heig-vd

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

Build solid REST APIs

Apply BDD with Swagger. Spring Boot and Cucumber

AMT 2018

Olivier Liechti

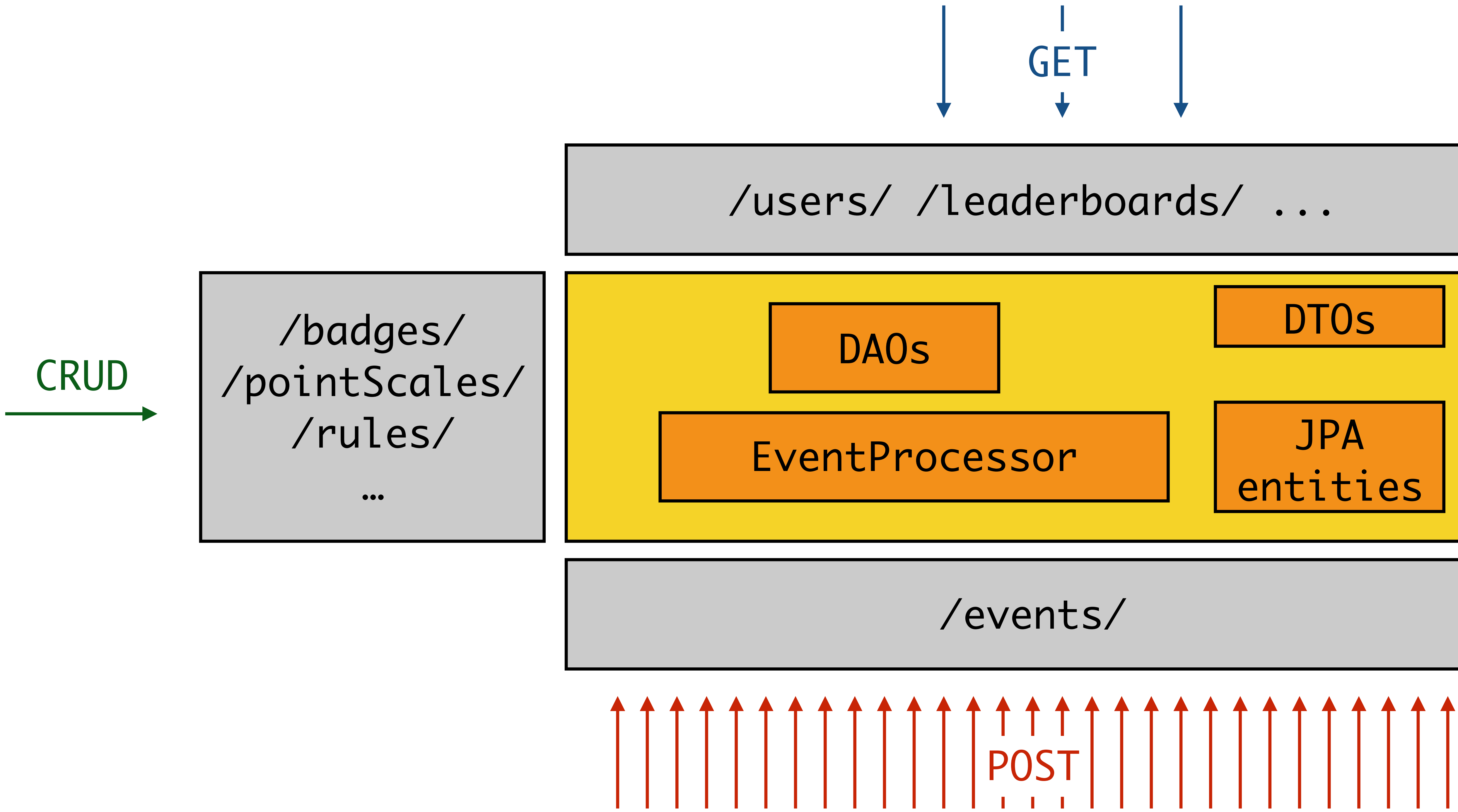
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Introduction

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

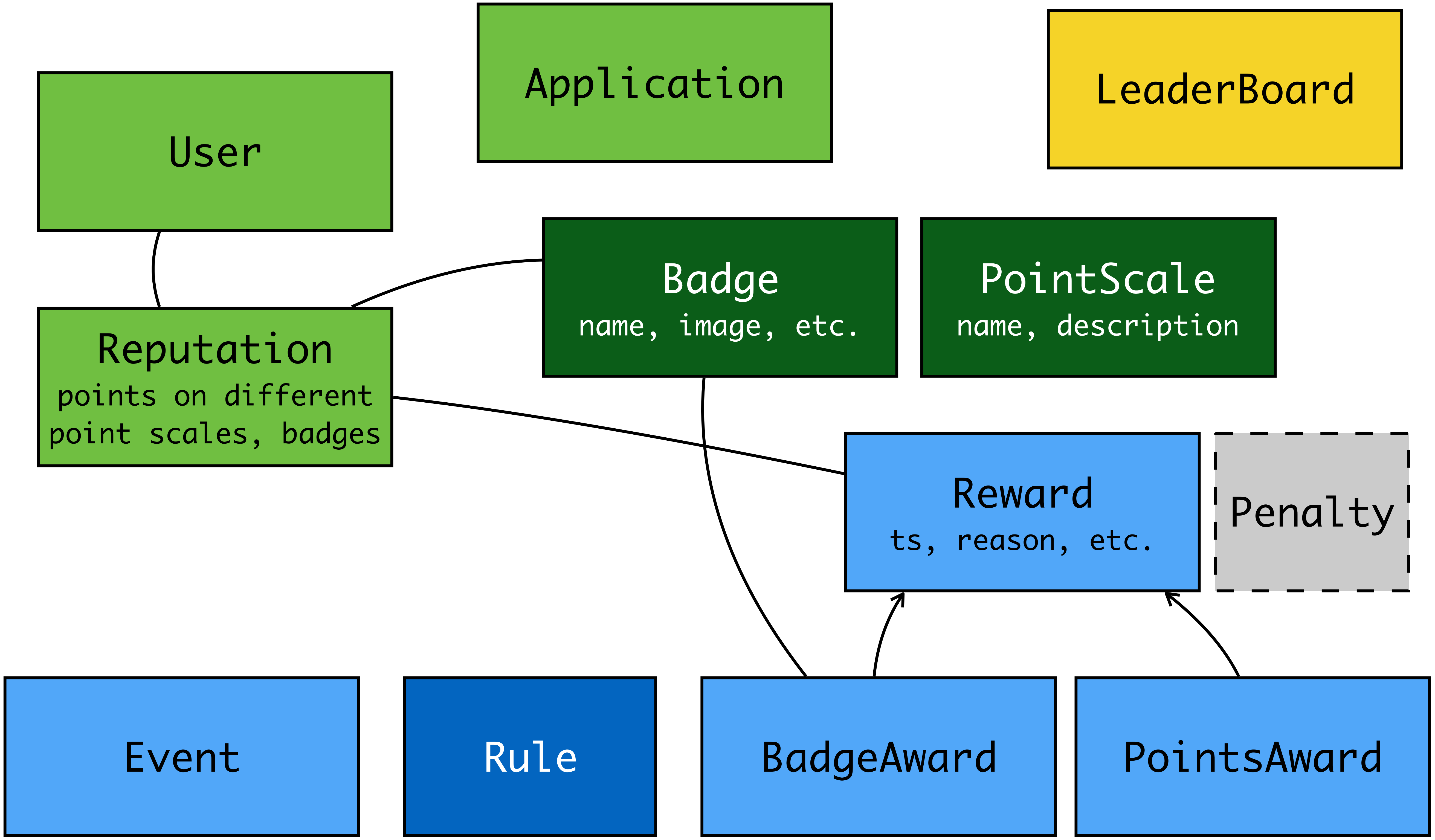
From…	**To…**
• Core Java EE APIs • Servlet/JSPs • JPA	• Higher-level frameworks • Spring MVC • Spring Data
• Dependency injection with the application server	• Dependency injection with Spring
• Server-side MVC	• REST APIs
• .war packages deployed in containers	• containers embedded in .jar executables

High-level architecture



Domain model (illustrative and partial)

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud



What is an event? (this is only a draft)

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

```
event : {
  userId: idInTheGamifiedApp,
  timestamp : 2018-12-17:17-00-00,
  type: drink,
  properties: {
    type: beer,
    quantity: some
  }
}
```

What is a rule? (this is only a draft)

```
rule : {
  if: {
    type: drink
  },
  then : {
    awardBadge : /badges/champion,
    awardPoints : {
       pointScale : /pointScales/health,
       amount: 1000
    }
  }
}
```

Authentication for REST endpoints

```
GET /badges HTTP/1.1
Accept: application/json
```

Who is calling me?

```
GET /badges HTTP/1.1
Accept: application/json
X-Api-Key: A83C-B99B-91VW-YZ1L
```

I return the badges created for this application

Schedule until Christmas

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

19.11.2018 Spring, Swagger, Cucumber	**23.11.2018** 15' tutorial on Spring Data BDD for `/badges` and `/pointScales`
26.11.2018 Travail écrit	**30.11.2018** BDD for `/events` Preparation of JMeter scripts
3.12.2018 Spring Data behind the scenes + selected topics	**7.12.2018** BDD for /rules
10.12.2018 Design of event processing service	**14.12.2018** Load tests & documentation
17.12.2018 Pré-raclette	**21.12.2018** Raclette

Schedule for today

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

15:45 - 16:00	Intro	gamification (10') schedule (5')
16:00 - 16:30	First steps with Spring Boot	intro (10') tutorial (20')
16:30 - 17:30	REST APIs with Swagger	exercise (15' + 15') demo
17:30 - 18:00	BDD with CucumberJVM	intro (10') demo (20')
18:00 - 18:05	Project	Next steps

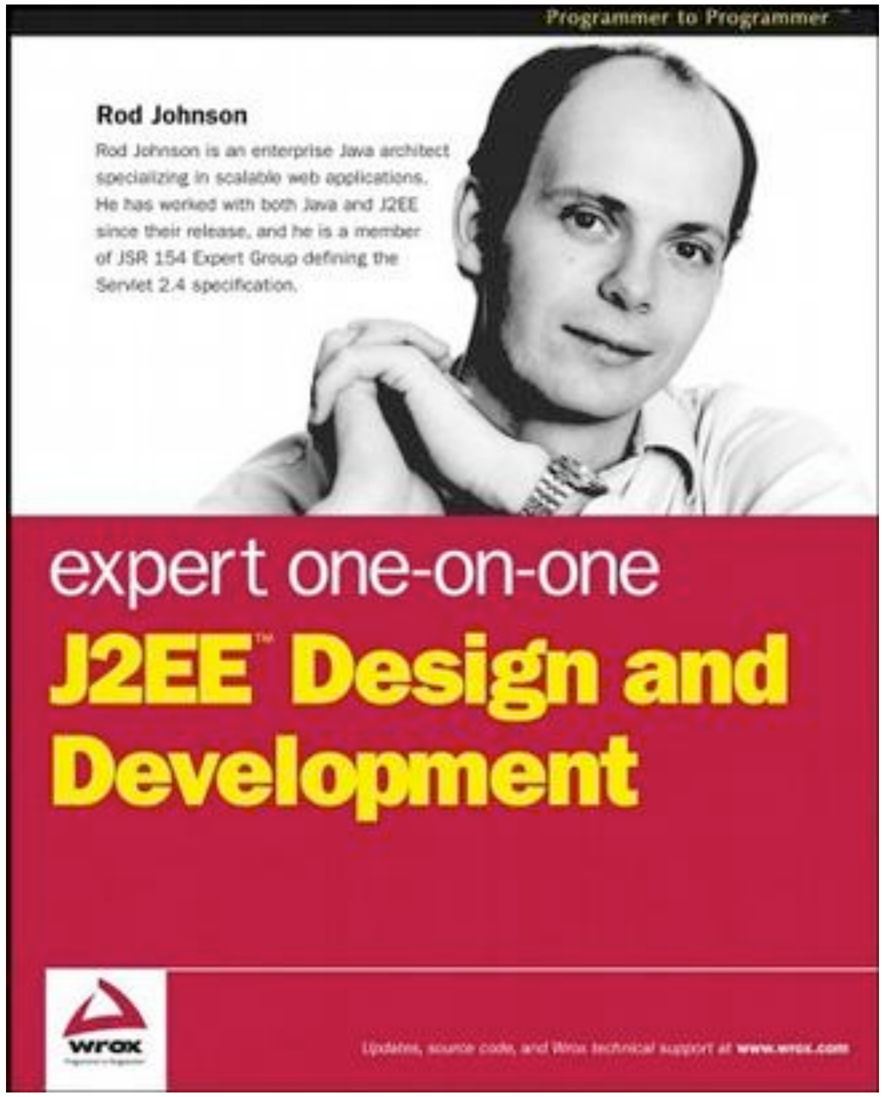
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

First steps with Springboot

The Spring Framework

- **When was it developed?**
  - The Spring Framework was released in 2003.
  - It was developed by Rod Johnson and presented in the book "Expert One-on-One J2EE Design and Development".
  - The framework has quickly become very popular and has expanded a lot since its inception (also through "acquisitions" of open source projects)
- **Why was it developed?**
  - The Spring Framework was developed at the time of J2EE and EJB 2.
  - At the time, using Enterprise Java Beans was rather "painful".
  - The Spring Framework proposed a lightweight approach, which was appropriate in many situations (for which J2EE was overkill).

Rod Johnson



Programmer to Programmer™

Rod Johnson

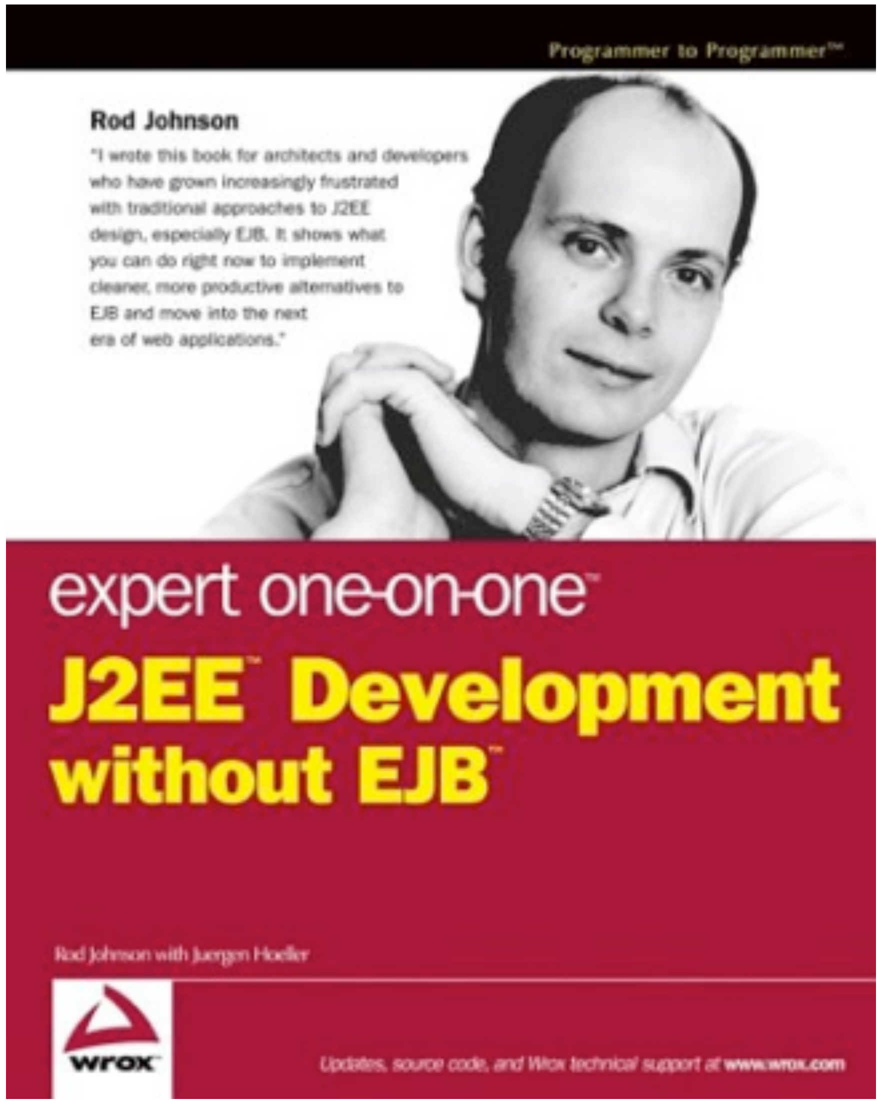
Rod Johnson is an enterprise Java architect specializing in scalable web applications. He has worked with both Java and J2EE since their release, and he is a member of JSR 154 Expert Group defining the Servlet 2.4 specification.

expert one-on-one

J2EE™ Design and Development

wrox

Updates, source code, and Wrox technical support at www.wrox.com



Programmer to Programmer™

Rod Johnson

"I wrote this book for architects and developers who have grown increasingly frustrated with traditional approaches to J2EE design, especially EJB. It shows what you can do right now to implement cleaner, more productive alternatives to EJB and move into the next era of web applications."

expert one-on-one™

J2EE™ Development without EJB™

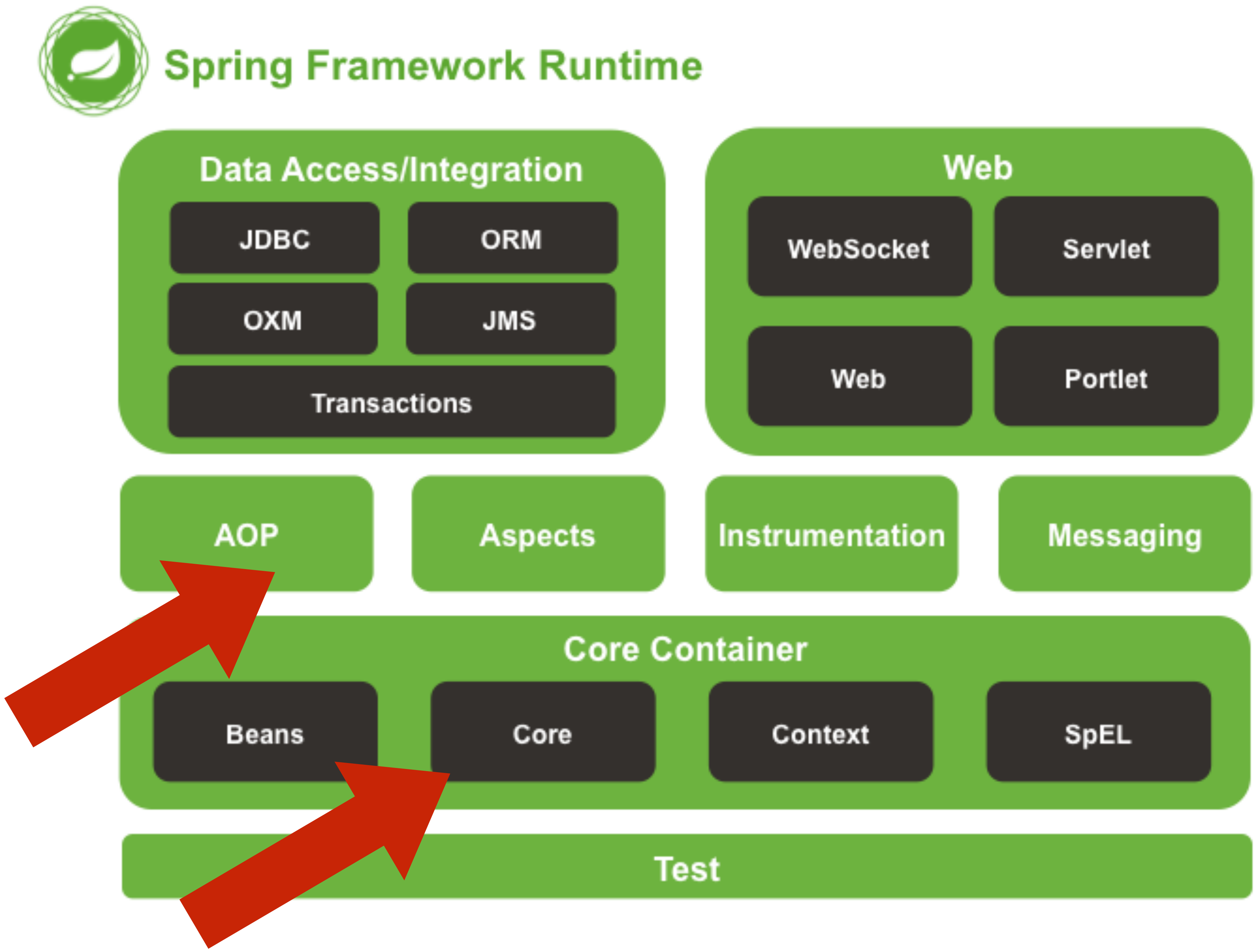
Rod Johnson with Juergen Hoeller

wrox

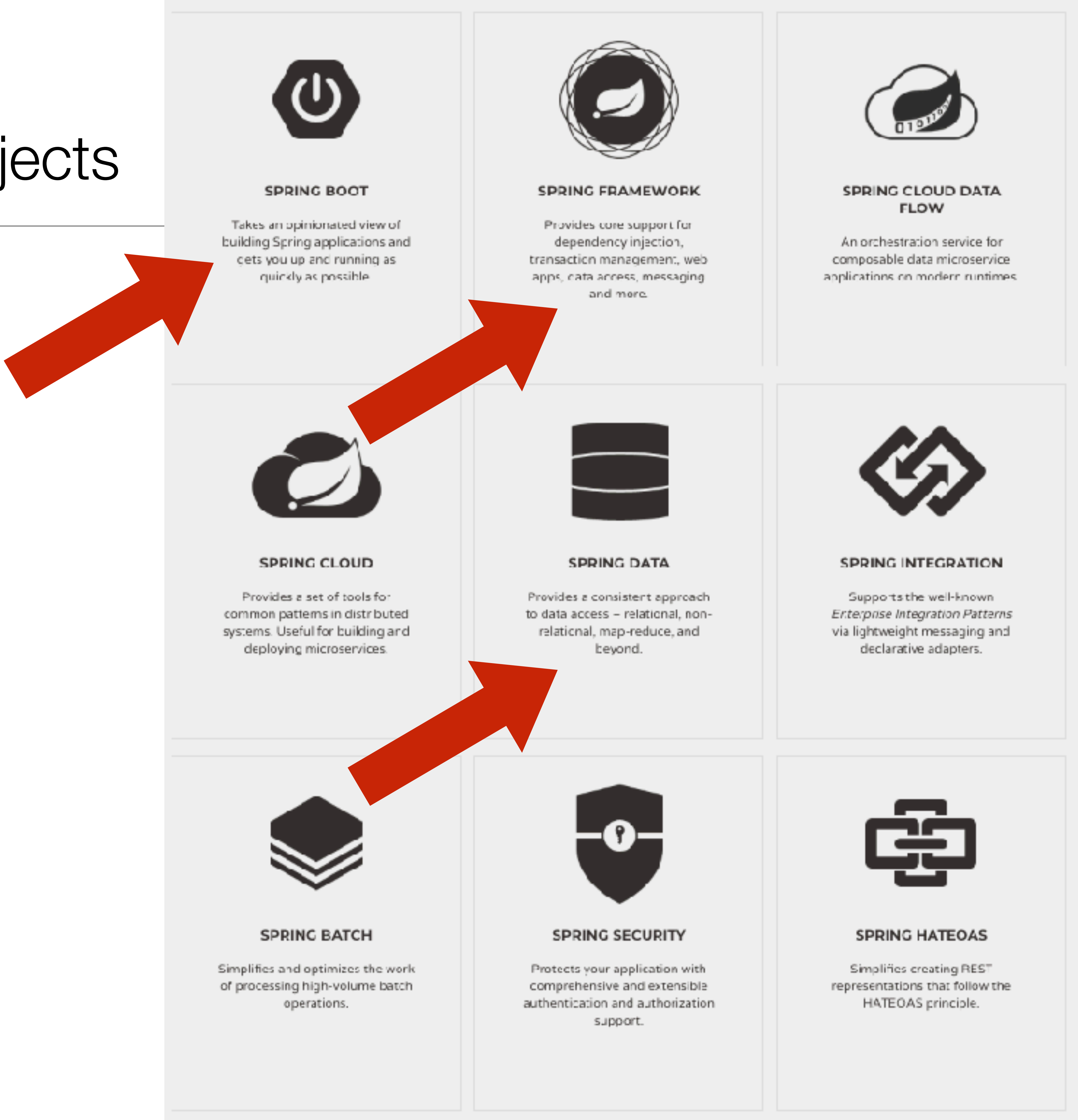
Updates, source code, and Wrox technical support at www.wrox.com

Spring Framework

- Spring enables you to build applications from POJOs and to apply enterprise services non-invasively.
- This capability applies to the Java SE programming model and to full and partial Java EE.



Spring.io Projects



SPRING BOOT

Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

SPRING FRAMEWORK

Provides core support for dependency injection, transaction management, web apps, data access, messaging and more.

SPRING CLOUD DATA FLOW

An orchestration service for composable data microservice applications on modern runtimes.

SPRING CLOUD

Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.

SPRING DATA

Provides a consistent approach to data access – relational, non-relational, map-reduce, and beyond.

SPRING INTEGRATION

Supports the well-known *Enterprise Integration Patterns* via lightweight messaging and declarative adapters.

SPRING BATCH

Simplifies and optimizes the work of processing high-volume batch operations.

SPRING SECURITY

Protects your application with comprehensive and extensible authentication and authorization support.

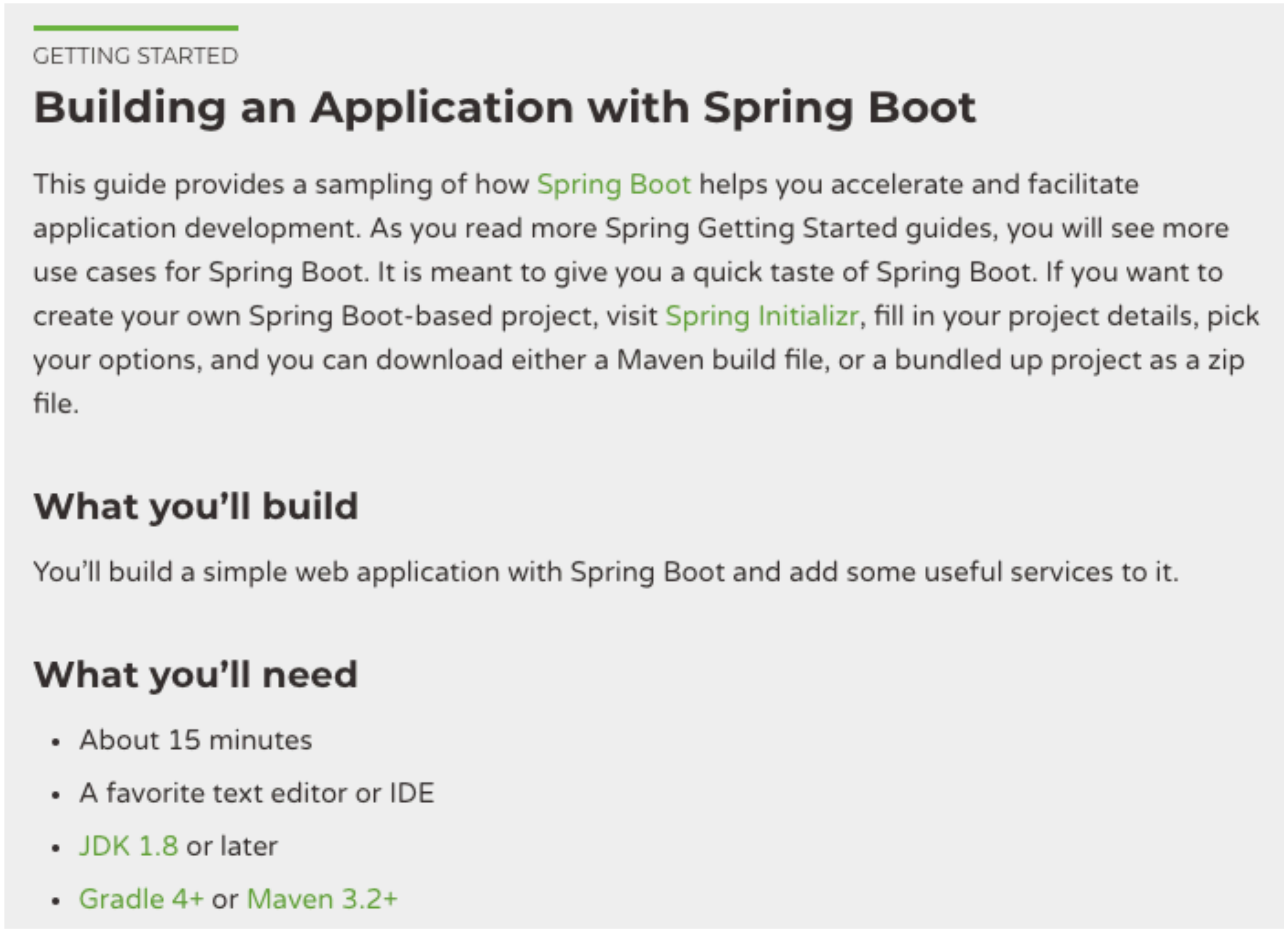
SPRING HATEOAS

Simplifies creating REST representations that follow the HATEOAS principle.

Spring Boot in practice

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

https://spring.io/guides/gs/spring-boot/



GETTING STARTED

Building an Application with Spring Boot

This guide provides a sampling of how Spring Boot helps you accelerate and facilitate application development. As you read more Spring Getting Started guides, you will see more use cases for Spring Boot. It is meant to give you a quick taste of Spring Boot. If you want to create your own Spring Boot-based project, visit Spring Initializr, fill in your project details, pick your options, and you can download either a Maven build file, or a bundled up project as a zip file.

What you'll build

You'll build a simple web application with Spring Boot and add some useful services to it.

What you'll need

- About 15 minutes
- A favorite text editor or IDE
- JDK 1.8 or later
- Gradle 4+ or Maven 3.2+

use IDEA

pick maven

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

REST APIs with Swagger

REST APIs

Everybody has already used and implemented a REST API (initially, maybe without having heard this acronym).

Simple REST endpoints expose (some of the) CRUD methods. You know that. Just don't feel obliged to implement every CRUD method (and assess the implications, in particular for DELETE).

But with rich domain models, you should not simply do a CRUD interface for every business entity. You also have to think about workflows and actions. Think about recording events that trigger state transitions.

Best practices

https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9

https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api

https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

Introductory exercise

- *Design a REST API, so that it is possible to implement the following stories:*
    - *As an **HR admin**, I can create employees.*
    - *As an **HR admin**, I can retrieve the list of employees (also the employees who report to a certain manager).*
    - *As an **employee**, I can make a request to take a vacation. I need to indicate the start and end dates, as well as a short description.*
    - *As a **manager**, I can see the list of the pending requests that <u>I</u> can and need to process. I can approve or reject requests.*
    - *As an employee, I can see the **status** of <u>my</u> requests.*

Questions

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- What are the **resources** in the application?
- What **URLs** should we use in our API?
- How do we model **actions** (make, approve, reject, etc.)
- How do we deal with **lists** and **pagination**?
- How do we deal with **linked resources** (e.g. *employee-requests*)
- How do we deal with **identification, authentication** and **authorization**?

API Spec

- The API Spec is defined by:
  - URLs
  - Methods allowed on each URL and their semantics
  - Payloads (both for requests and responses)
  - Parameters in the query string and in HTTP headers

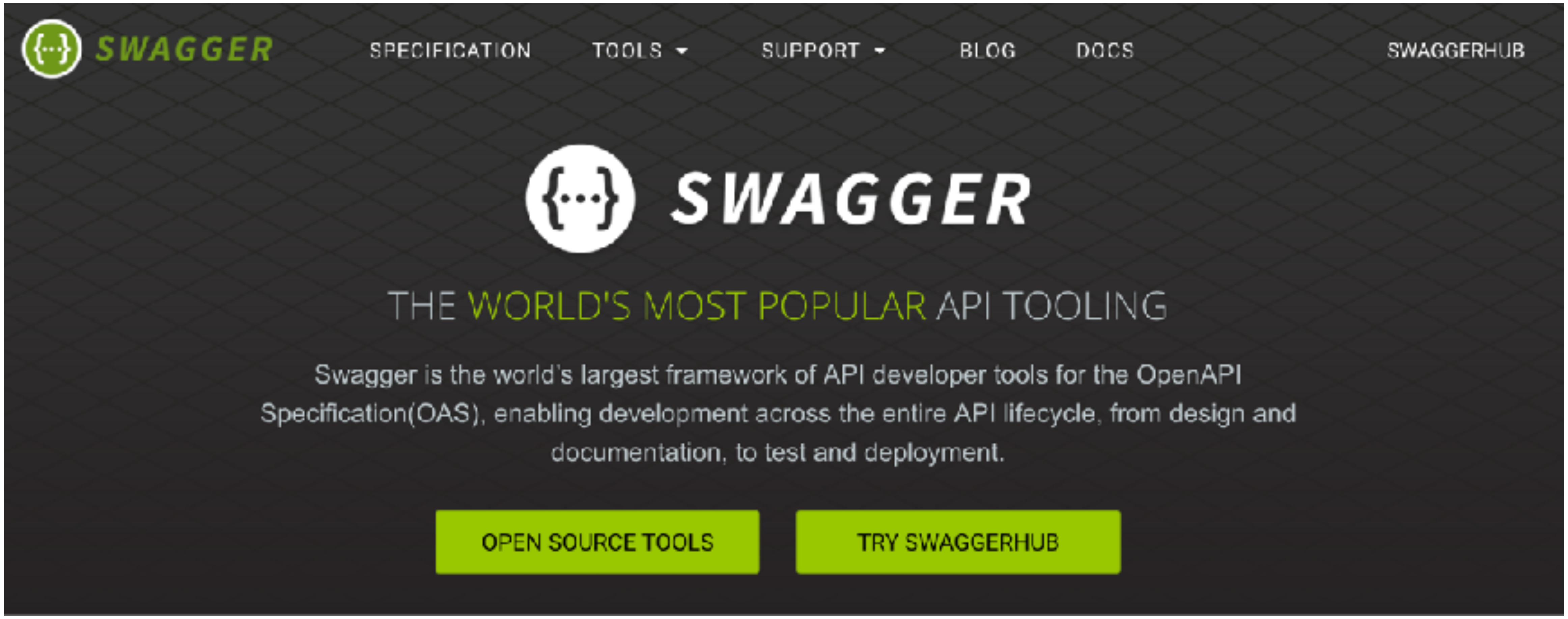
Take 15' minutes to sketch the HR API
We will take 15' to review some of your proposals

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Getting started with Swagger

API specification with Swagger

heig-vd HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD www.heig-vd.ch

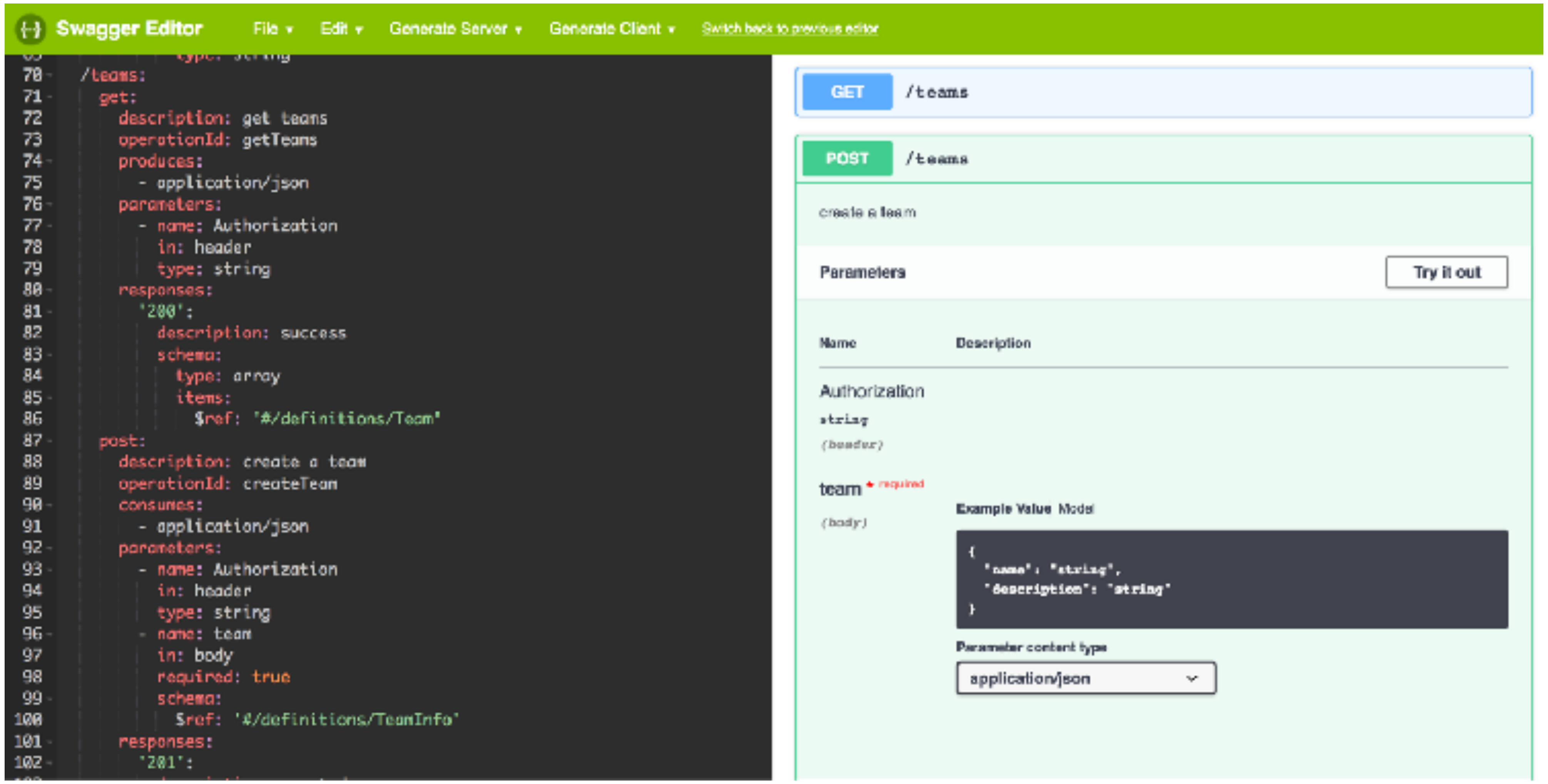


THE WORLD'S MOST POPULAR API TOOLING

Swagger is the world's largest framework of API developer tools for the OpenAPI Specification(OAS), enabling development across the entire API lifecycle, from design and documentation, to test and deployment.

Interactive documentation

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



```
        type: string
/teams:
  get:
    description: get teams
    operationId: getTeams
    produces:
      - application/json
    parameters:
      - name: Authorization
        in: header
        type: string
    responses:
      '200':
        description: success
        schema:
          type: array
          items:
            $ref: '#/definitions/Team'
  post:
    description: create a team
    operationId: createTeam
    consumes:
      - application/json
    parameters:
      - name: Authorization
        in: header
        type: string
      - name: team
        in: body
        required: true
        schema:
          $ref: '#/definitions/TeamInfo'
    responses:
      '201':
```

GET /teams

POST /teams

create a team

Parameters — Try it out

Name	Description
Authorization string *(header)*	
team * required *(body)*	Example Value Model

```
{
  "name": "string",
  "description": "string"
}
```

Parameter content type: application/json

Top-Down

code generation

VS

Bottom-Up

annotations

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

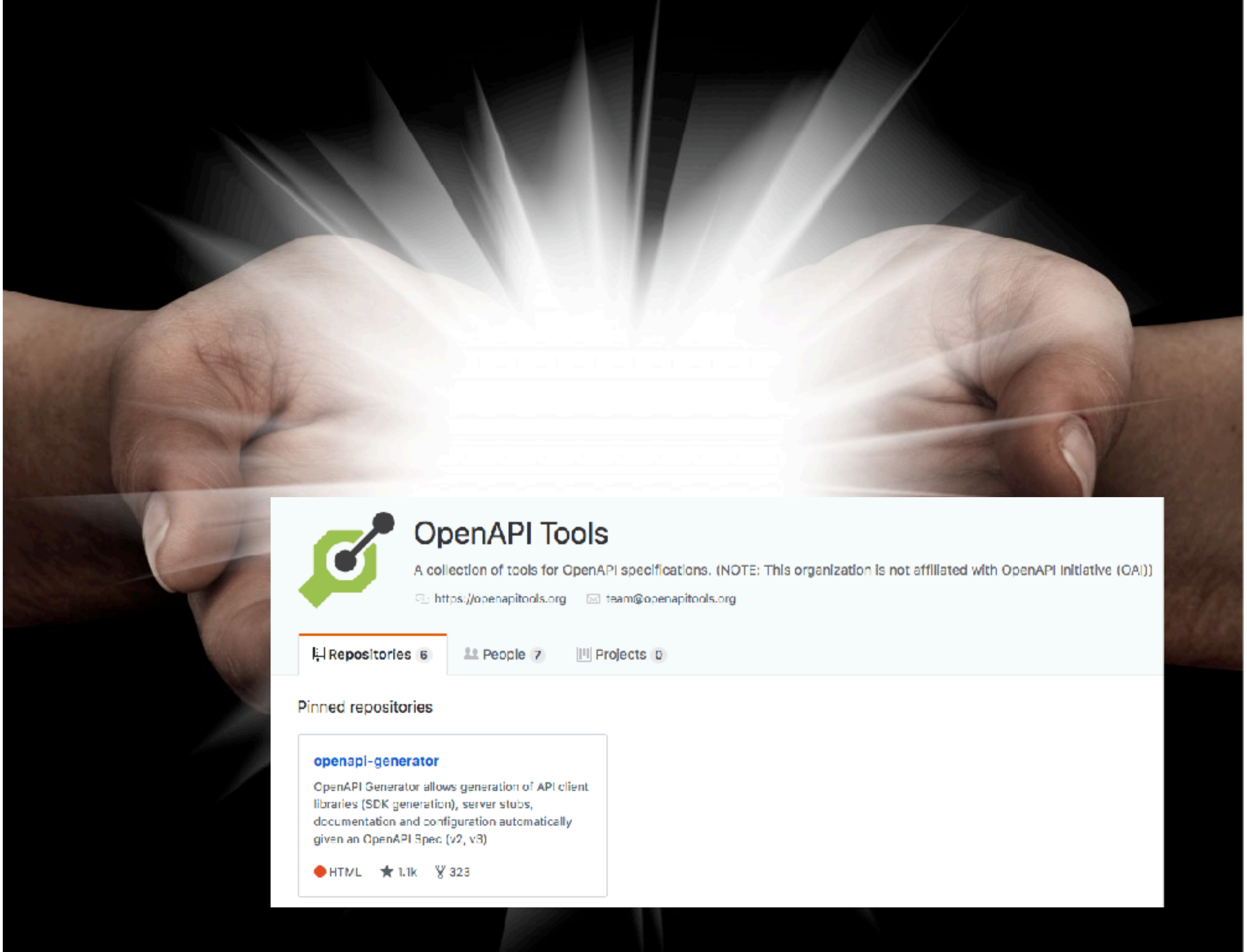
Editor v2

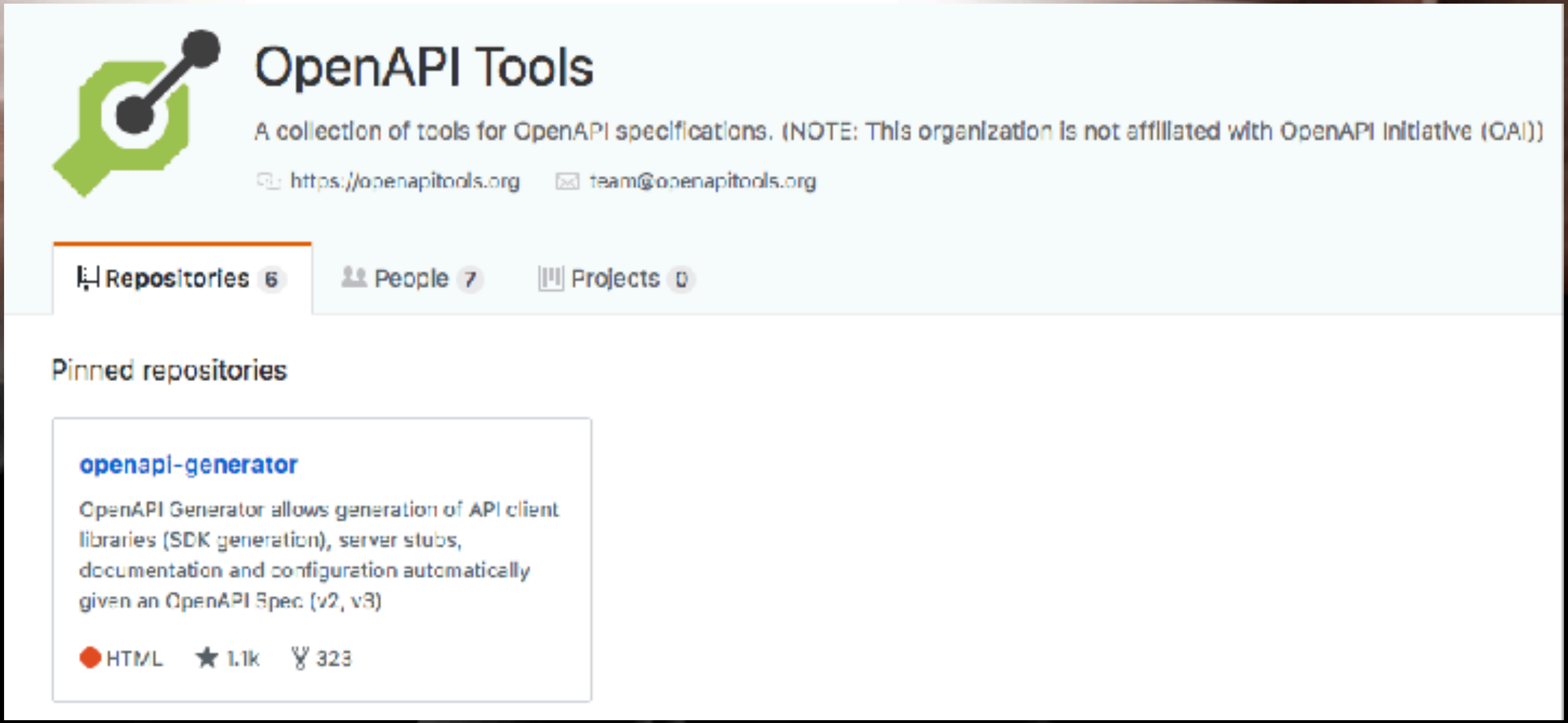
vs

Editor v3

vs

codegen





OpenAPI Tools

A collection of tools for OpenAPI specifications. (NOTE: This organization is not affiliated with OpenAPI Initiative (OAI))

https://openapitools.org team@openapitools.org

Pinned repositories

openapi-generator

OpenAPI Generator allows generation of API client libraries (SDK generation), server stubs, documentation and configuration automatically given an OpenAPI Spec (v2, v3)

● HTML ★ 1.1k 323

Current status

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- The projects that we provide in our repos still use the Swagger 2.0 specs (vs OpenAPI 3.0)
- We have solved a lot of issues and designed a development workflow. Use our pom.xml and project structure and you have something stable to work with.
- The GitHub organization "swagger-api" used to be the place where to get the tools. Be ready to read issues and build plugins yourself.
- For various reasons, the community has forked. We have limited experience with the new OpenAPITools GitHub organization. But this seems to a better maintained project.
- We will move from "swagger codegen" to "openapi-generator".

heig-vd HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

Step 1: describe

Let's look at an example

- **Clone our repo**: https://github.com/AvaliaSystems/TrainingREST
  - Checkout the **swagger-intro branch**
  - Open the **./swagger/examples/fruits-api.yml** file, copy content

- Open the **Swagger Editor v2**: http://editor2.swagger.io, paste content
- Read the **specification**, look at the interactive **documentation**

Resources, operations and types

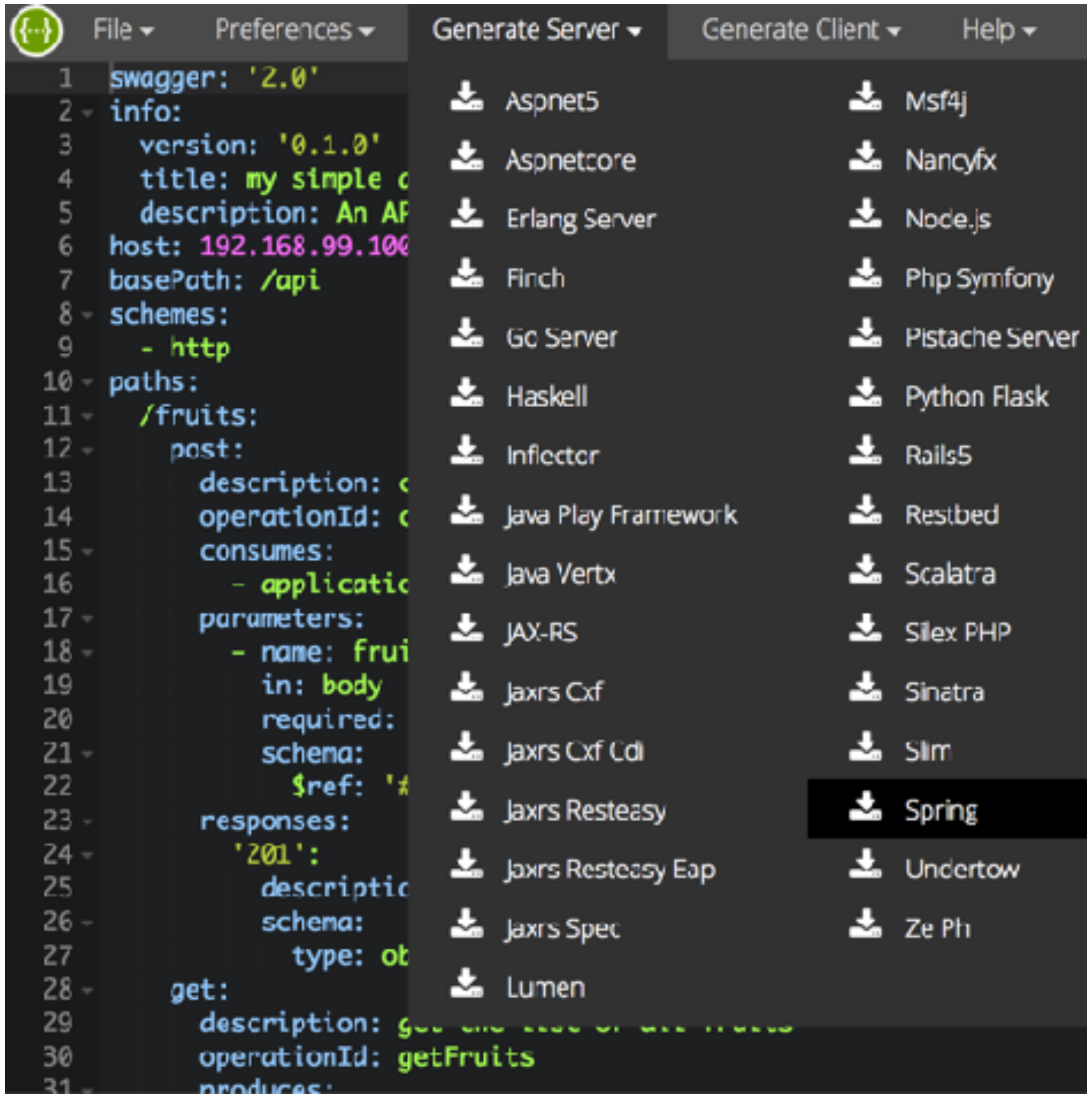
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
paths:
  /fruits:
    post:
      description: create a fruit
      operationId: createFruit
      consumes:
        - application/json
      parameters:
        - name: fruit
          in: body
          required: true
          schema:
            $ref: '#/definitions/Fruit'
      responses:
        '201':
          description: created
          schema:
            type: object
```

```
definitions:
  Fruit:
    type: object
    properties:
      kind:
        type: string
      colour:
        type: string
      size:
        type: string
```

heig-vd HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Step 2: implement



```
swagger: '2.0'
info:
  version: '0.1.0'
  title: my simple c
  description: An AP
host: 192.168.99.100
basePath: /api
schemes:
  - http
paths:
  /fruits:
    post:
      description: c
      operationId: c
      consumes:
        - applicatio
      parameters:
        - name: frui
          in: body
          required:
          schema:
            $ref: '#
      responses:
        '201':
          descriptio
          schema:
            type: ob
    get:
      description: get the list of all fruits
      operationId: getFruits
      produces:
```

```
spring-server
├── README.md
├── pom.xml
└── src
    └── main
        ├── java
        │   └── io
        │       └── swagger
        │           ├── RFC3339DateFormat.java
        │           ├── Swagger2SpringBoot.java
        │           ├── api
        │           │   ├── ApiException.java
        │           │   ├── ApiOriginFilter.java
        │           │   ├── ApiResponseMessage.java
        │           │   ├── FruitsApi.java
        │           │   ├──
FruitsApiController.java
        │           │   └── NotFoundException.java
        │           ├── configuration
        │           │   ├── HomeController.java
        │           │   └──
SwaggerDocumentationConfig.java
        │           └── model
        │               └── Fruit.java
        └── resources
            └── application.properties

9 directories, 14 files
```

Let's generate Java from the the spec

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

- In the editor, go to "**Generate Server**", "**Spring**"
- Unzip the skeleton and open the project in your **IDE**
- Fix **dependencies** in the **pom.xml** file
- Configure the **maven plugin** in the **pom.xml** (depends on your IDE)
- **Run**, either from command line (mvn spring-boot:run) or the IDE.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</
artifactId>
    <!--<scope>provided</scope>-->
</dependency>
```

```
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</
artifactId>
    <configuration>
        <fork>true</fork>
    </configuration>
</plugin>
```

```
public class Fruit   {
  @JsonProperty("kind")
  private String kind = null;

  @JsonProperty("colour")
  private String colour = null;

  @JsonProperty("size")
  private String size = null;
...
}
```

https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-requestmapping

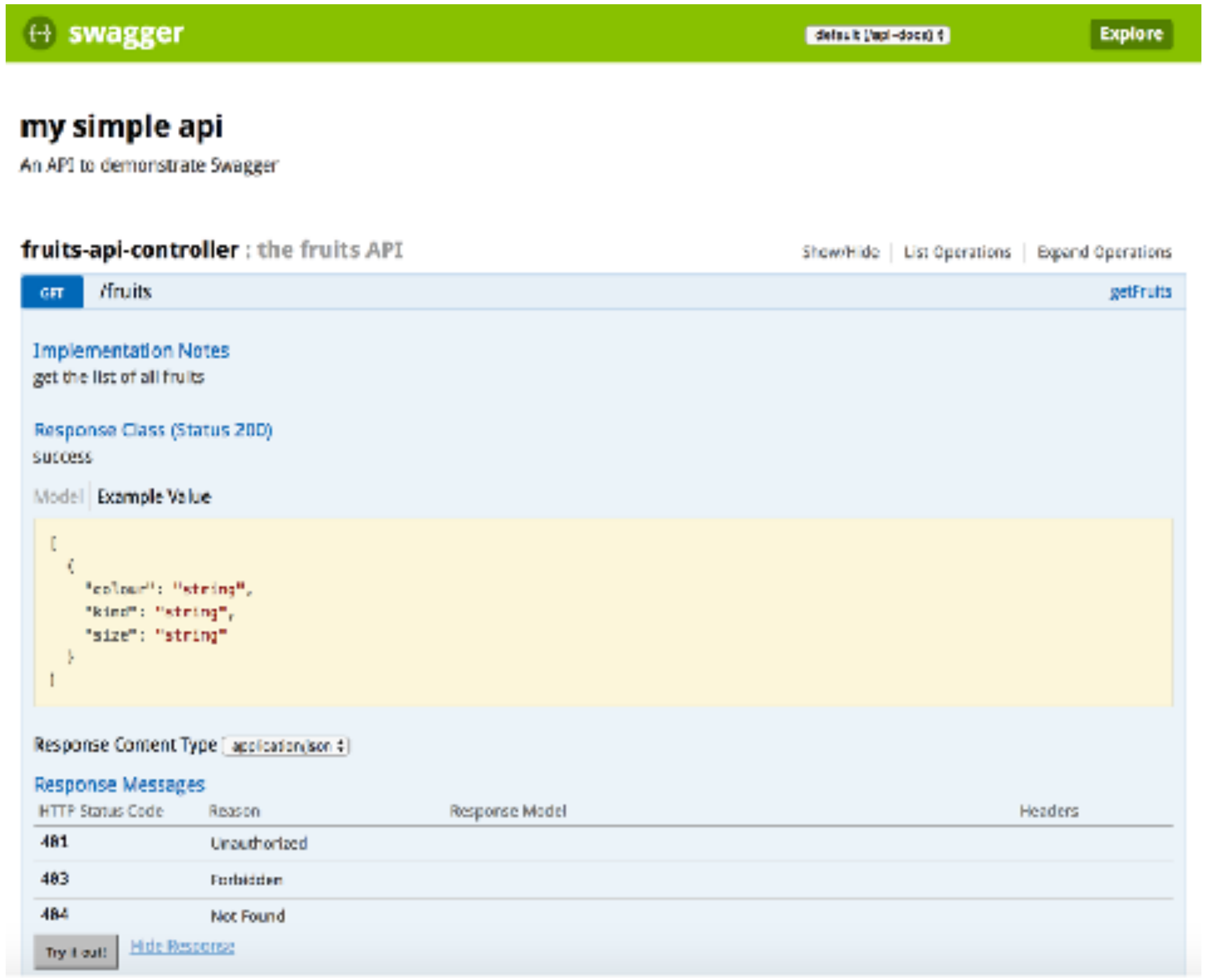
```
@Controller
public class FruitsApiController implements FruitsApi {

    public ResponseEntity<Object> createFruit(@ApiParam(value = "" ,required=true )  @Valid
@RequestBody Fruit fruit) {
        // do some magic!
        return new ResponseEntity<Object>(HttpStatus.OK);
    }

    public ResponseEntity<List<Fruit>> getFruits() {
        // do some magic!
        return new ResponseEntity<List<Fruit>>(HttpStatus.OK);
    }

}
```

Access documentation in the browser

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

- In the editor, go to "**Generate Server**", "**Spring**"
- Unzip the skeleton and open the project in your **IDE**
- Fix **dependencies** in the **pom.xml** file
- Configure the **maven plugin** in the **pom.xml** (depends on your IDE)

http://localhost:8080/api/swagger-ui.html



my simple api

An API to demonstrate Swagger

fruits-api-controller : the fruits API

Show/Hide | List Operations | Expand Operations

GET /fruits getFruits

Implementation Notes

get the list of all fruits

Response Class (Status 200)

success

Model | Example Value

```
[
  {
    "colour": "string",
    "kind": "string",
    "size": "string"
  }
]
```

Response Content Type application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Add persistence with Spring Data

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

- https://spring.io/guides/gs/accessing-data-jpa/
- Update dependencies in pom.xml
- Add a Fruit entity (DTO vs Entity!!)
- Add a Repository
- Inject dependency on Repository into API controller

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</
artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
</dependency>
```

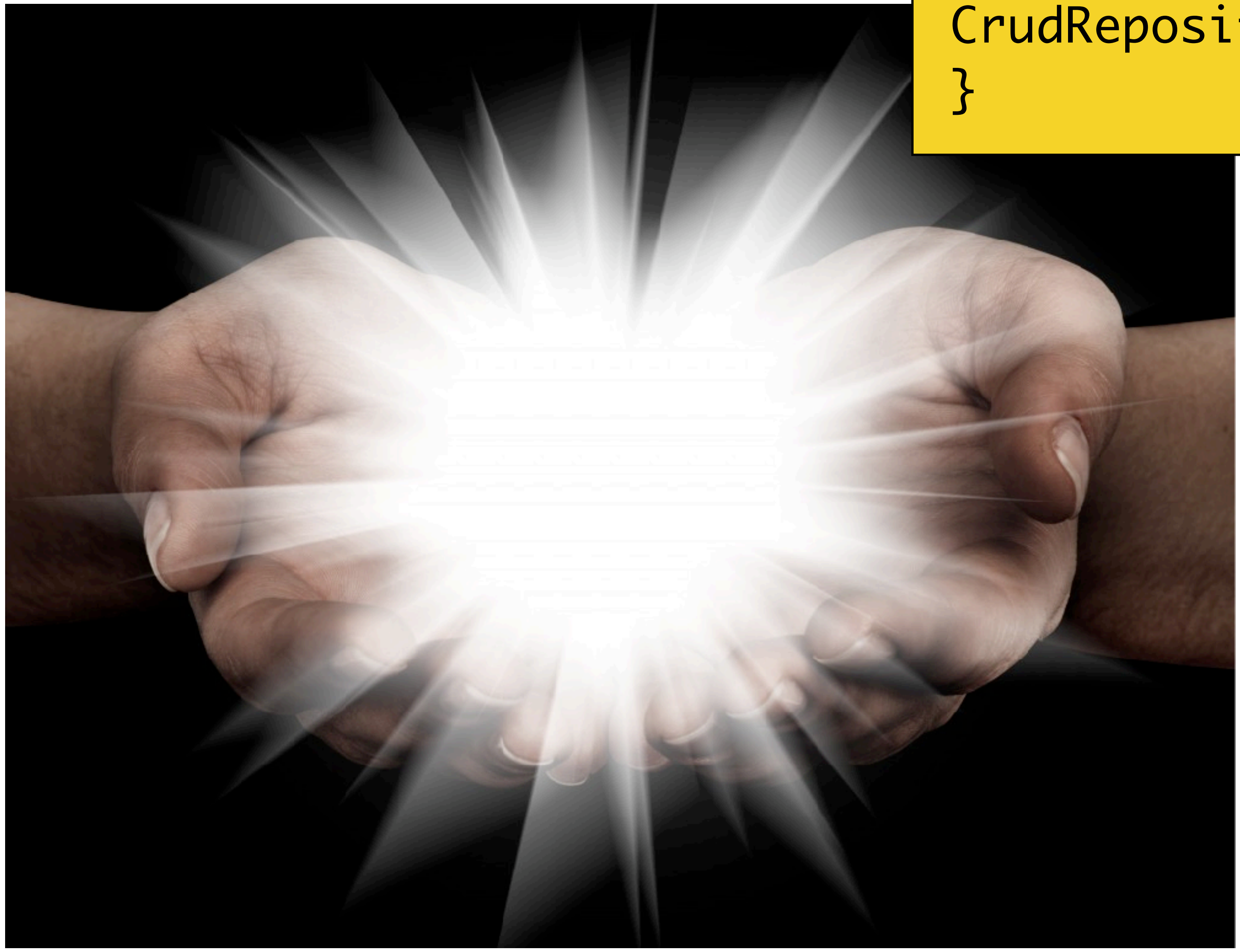
```
public interface FruitRepository extends
CrudRepository<FruitEntity, Long>{
}
```

```
@Entity
public class FruitEntity implements
Serializable {

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private long id;

    private String kind;
    private String size;
    private String colour;

    public long getId() {
        return id;
    }

    public String getKind() {
        return kind;
    }
...
}
```

Add persistence with Spring Data

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
public interface FruitRepository extends
CrudRepository<FruitEntity, Long>{
}
```

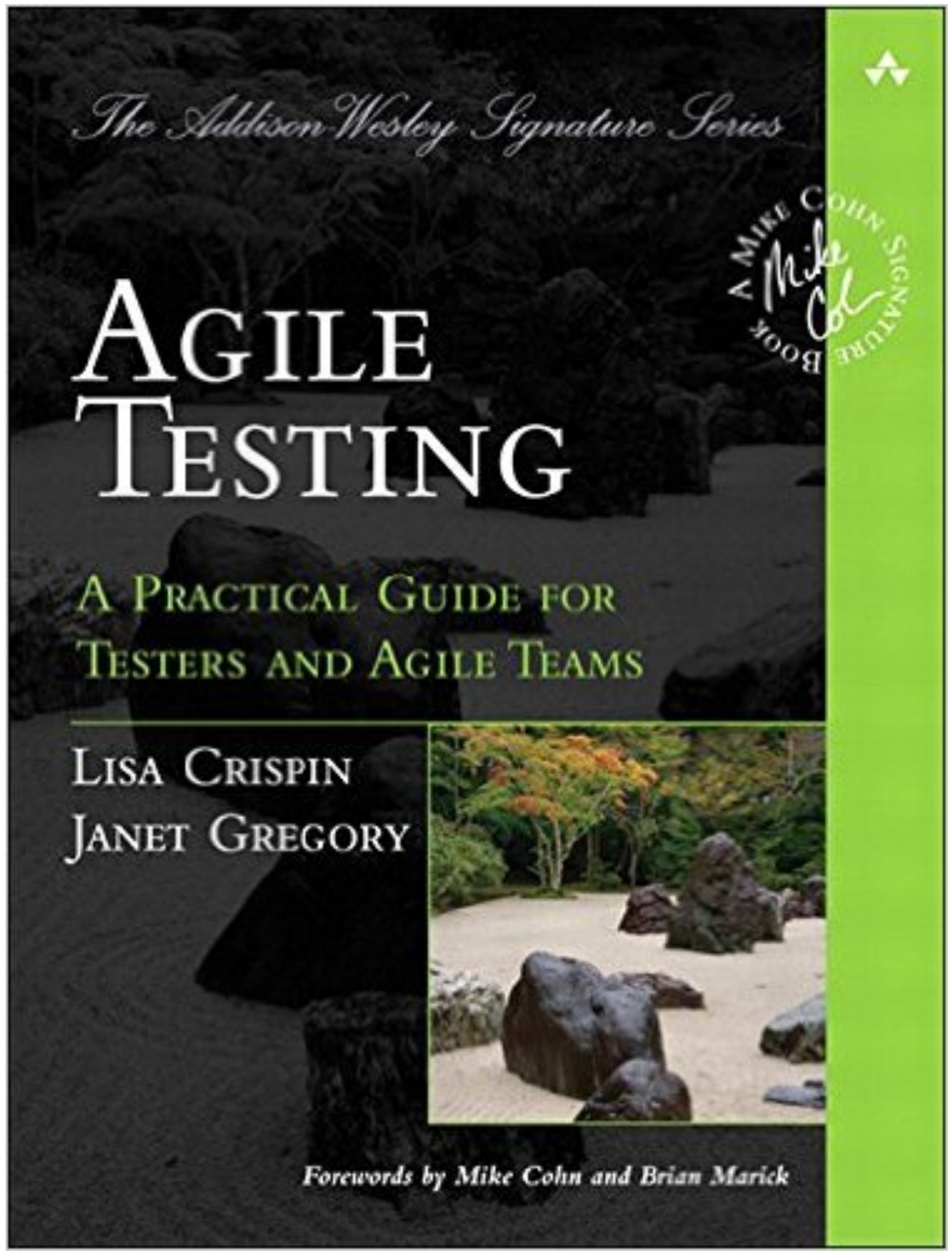


HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

BDD for REST APIs with CucumberJVM

*“**Software quality**” is a **broad concept** and has many aspects (reliability, efficiency, usability, maintainability, etc.).*

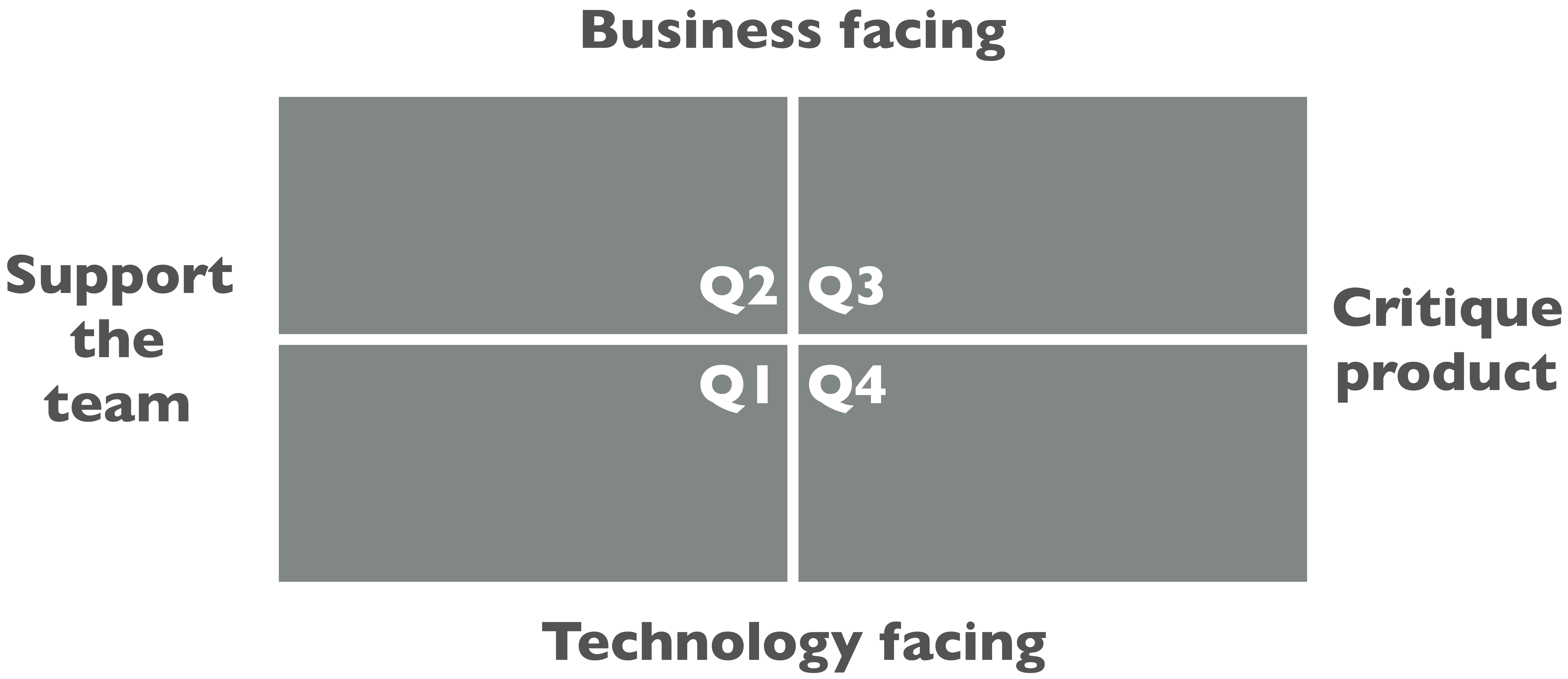
*“**Software testing**” refers to methods and techniques for **assessing** certain aspects of the quality of a software system. **There are many, many of them**.*


The Addison Wesley Signature Series
A Mike Cohn Signature Book
Agile Testing
A Practical Guide for Testers and Agile Teams
Lisa Crispin
Janet Gregory
Forewords by Mike Cohn and Brian Marick

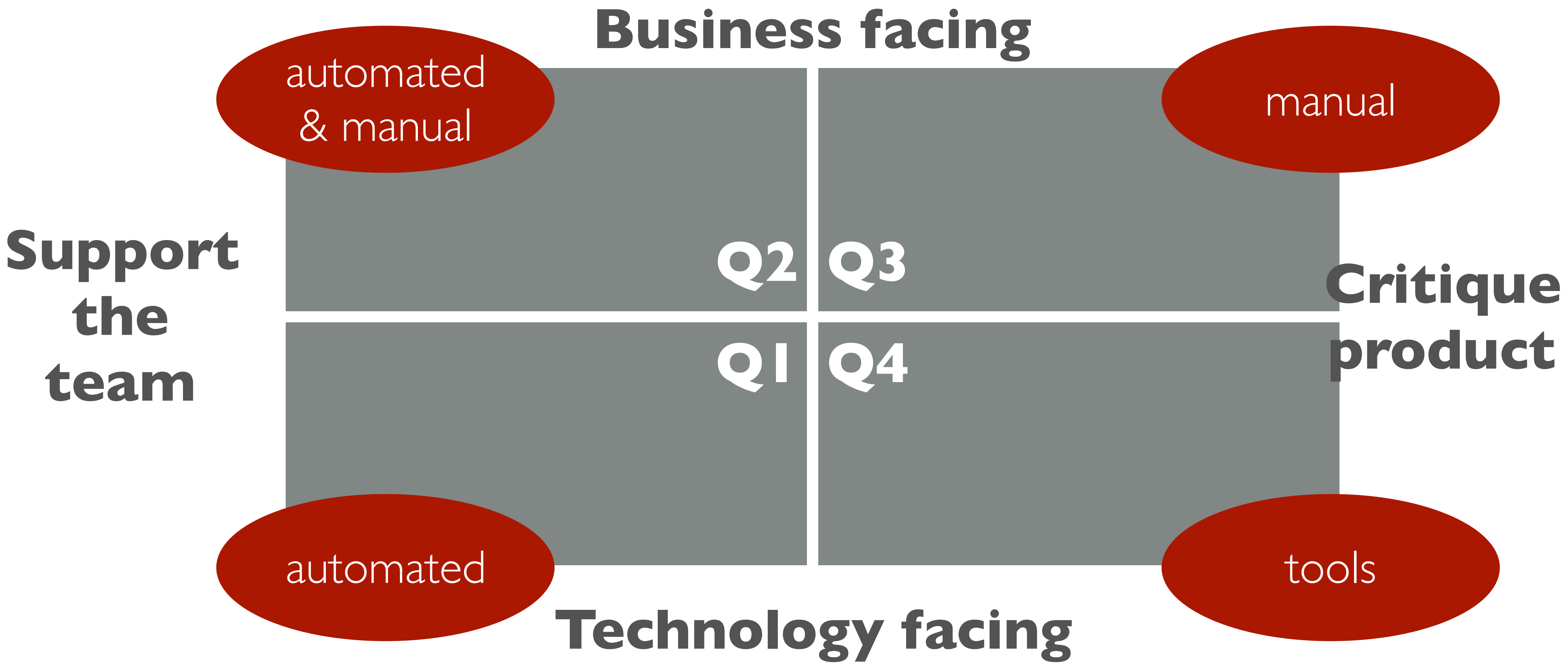
*Some “**Software testing**” techniques do not only measure quality after the fact, but **help the team to proactively** maintain the quality of the software to an appropriate level.*

*Is there a way to **classify** all these methods, so that we can see how they relate to each other?*

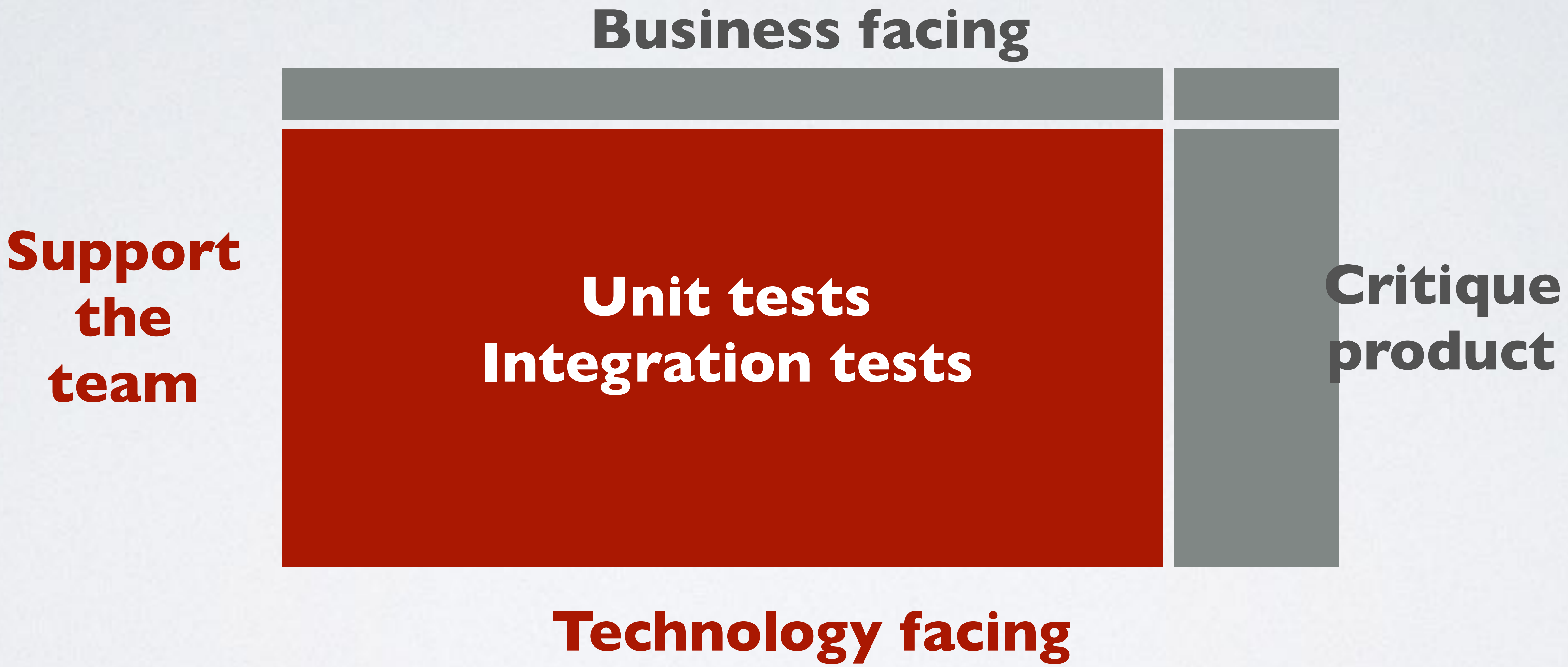
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



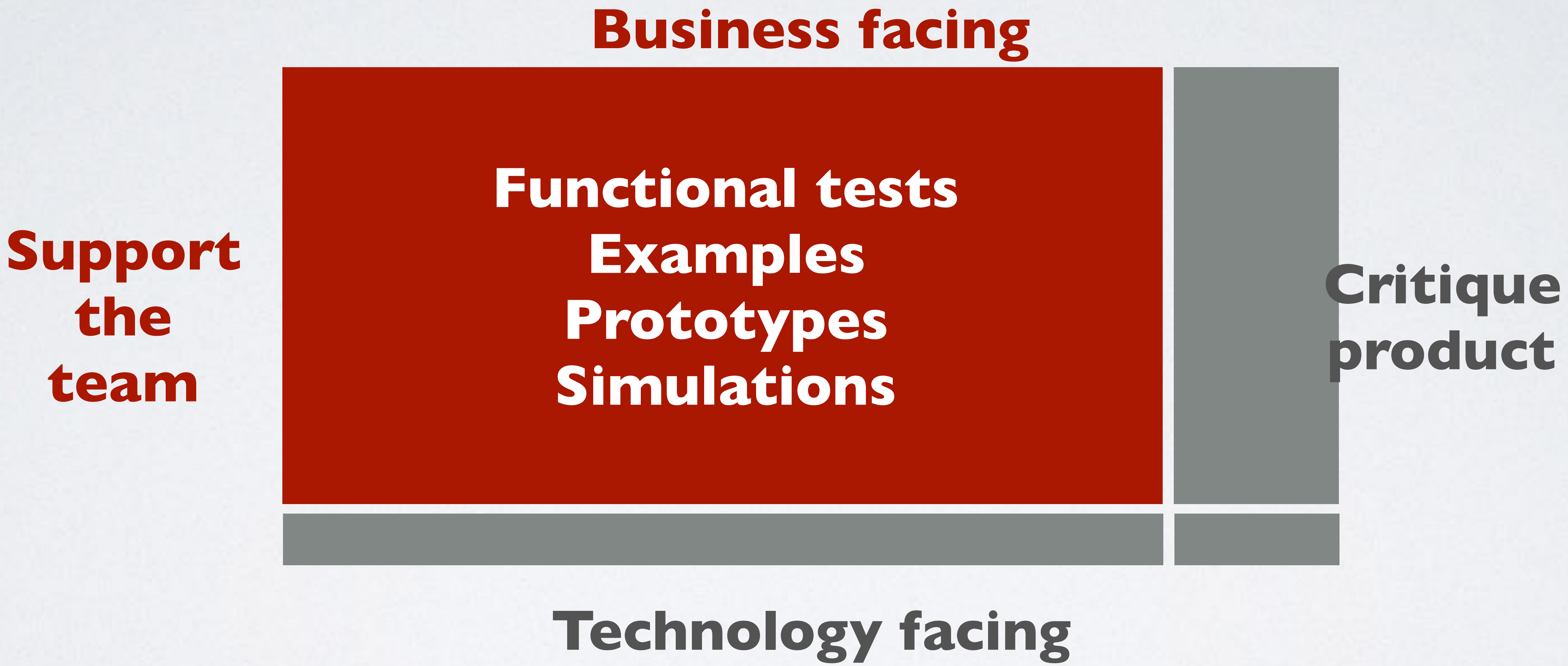
Support the team	Some of these tests **help individual team members** while they do their job. Sometimes, creating a “test” helps me **specify and/or design** the product. Other tests **facilitate team collaboration**, especially between “business” and “technical” people (shared language).
Critique product	Some of these tests allow humans to evaluate the quality of a software from the **users point of view** (is it easy to use? is it easy to learn? does it solve the user’s problem?). Other tests aim to detect issues with **non-functional (systemic) qualities**.
Technology facing	Some tests are created and executed by **technical team members**. They are highly automated. They relate to the “Are we building the product right?” question.
Business facing	Some tests are created by (or at least with) **business-oriented team members**. They also relate to the “Are we building the right product?” question.


Business facing
automated & manual
manual
Support the team
Q2
Q3
Q1
Q4
Critique product
automated
tools
Technology facing

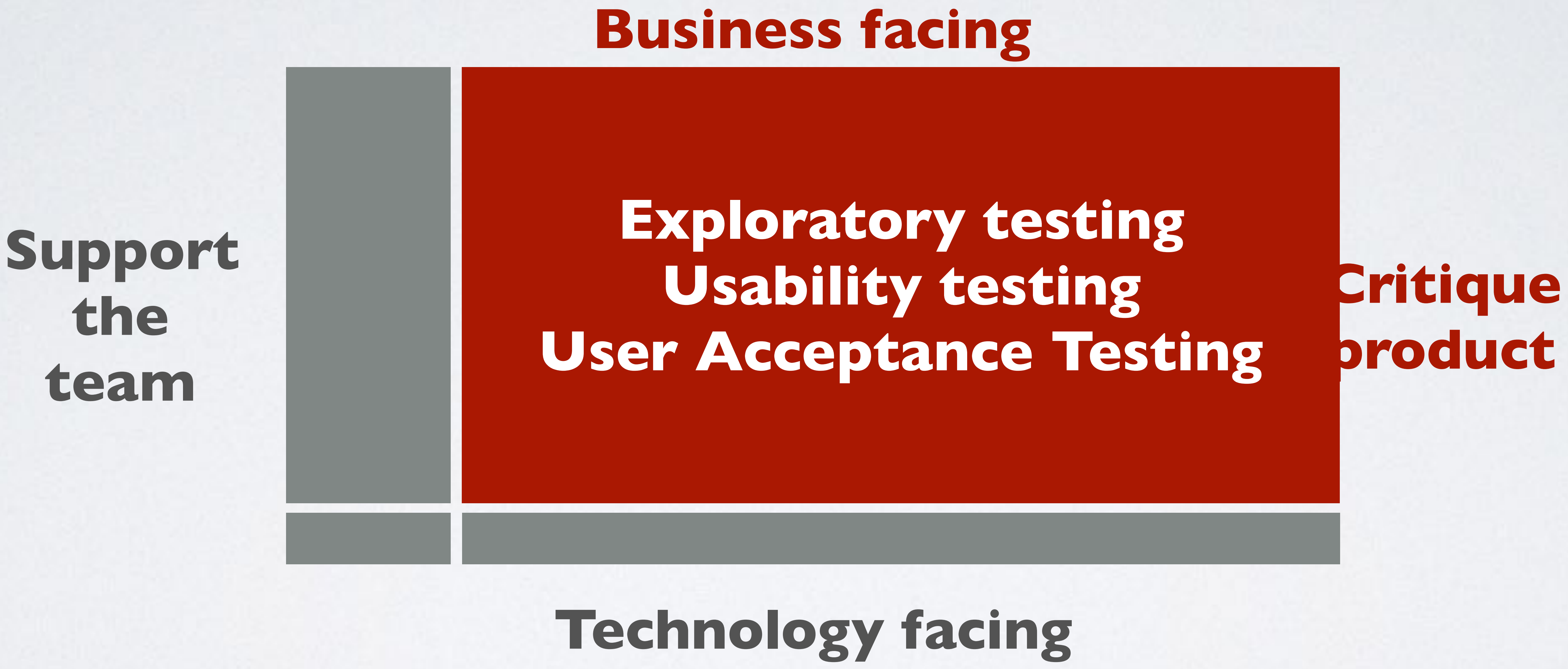
AGILE TESTING QUADRANTS: Q1



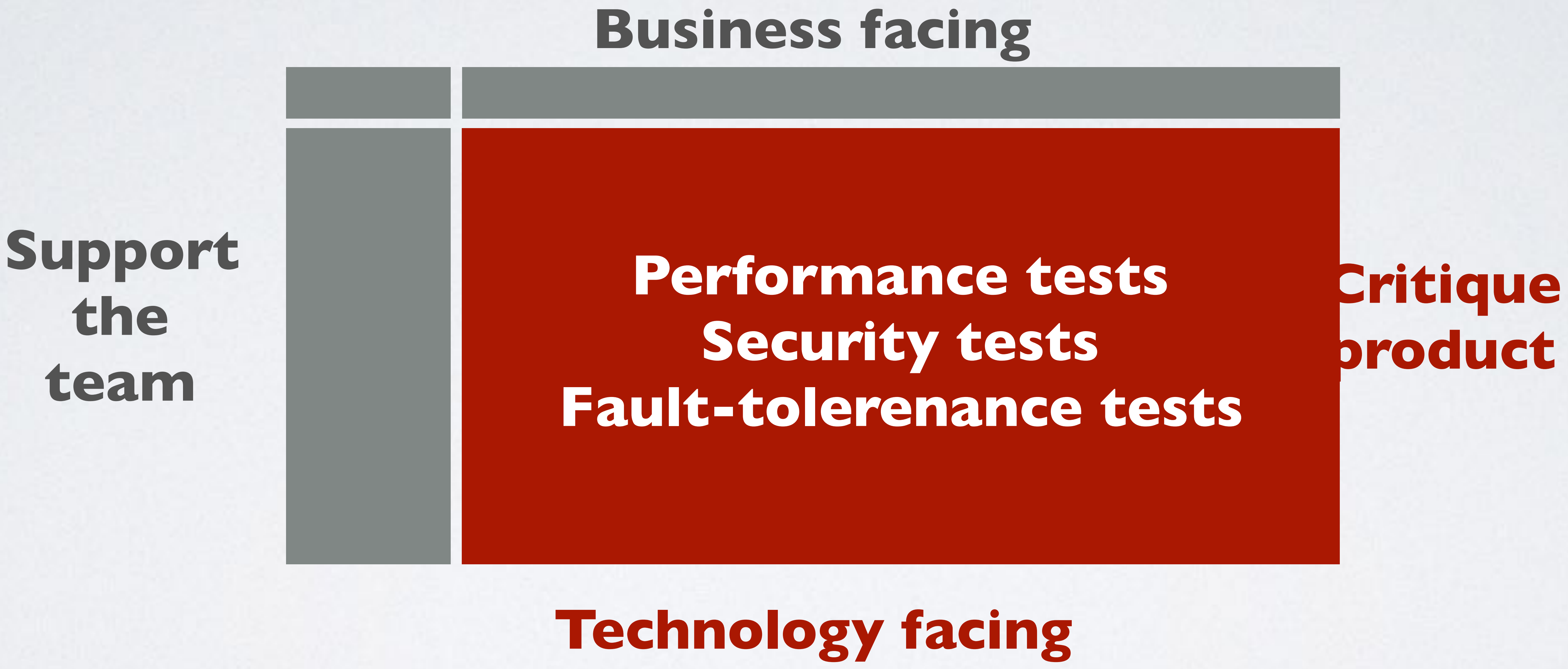
AGILE TESTING QUADRANTS: Q2



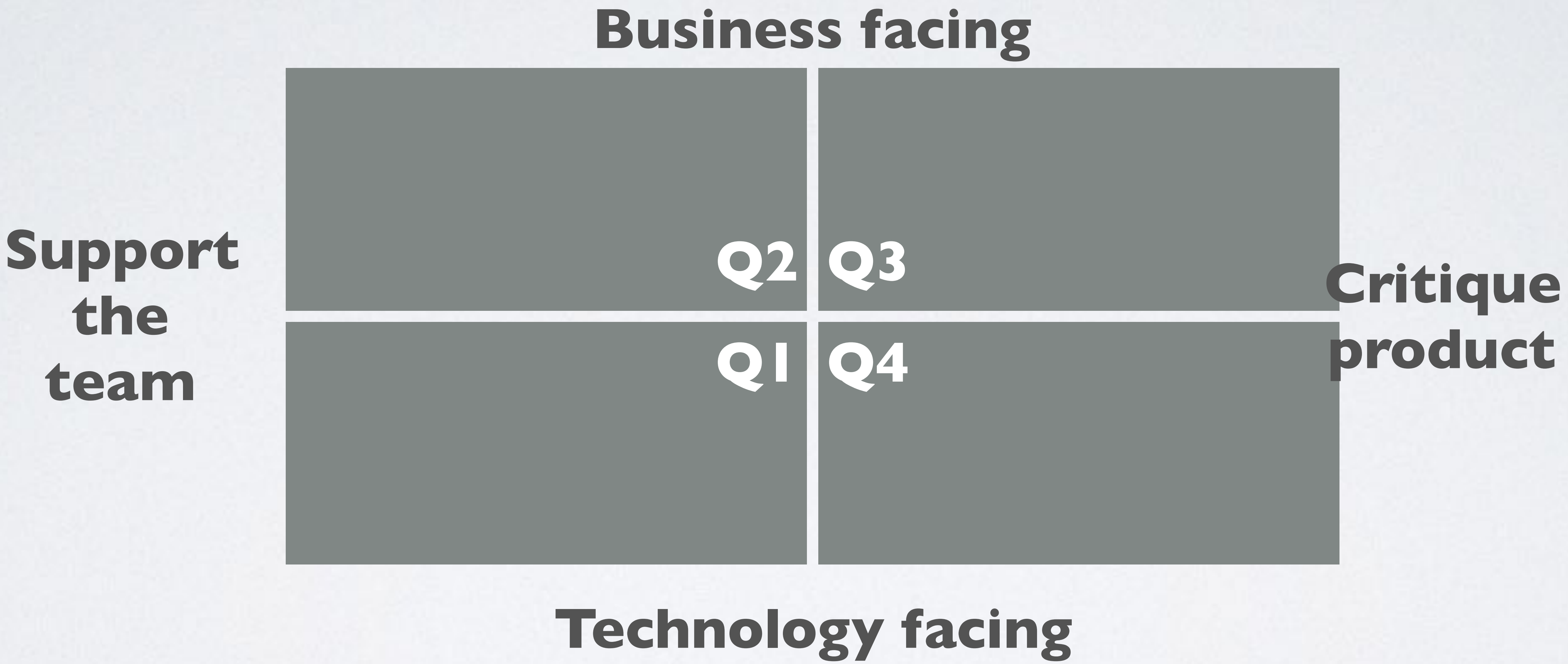
AGILE TESTING QUADRANTS: Q3



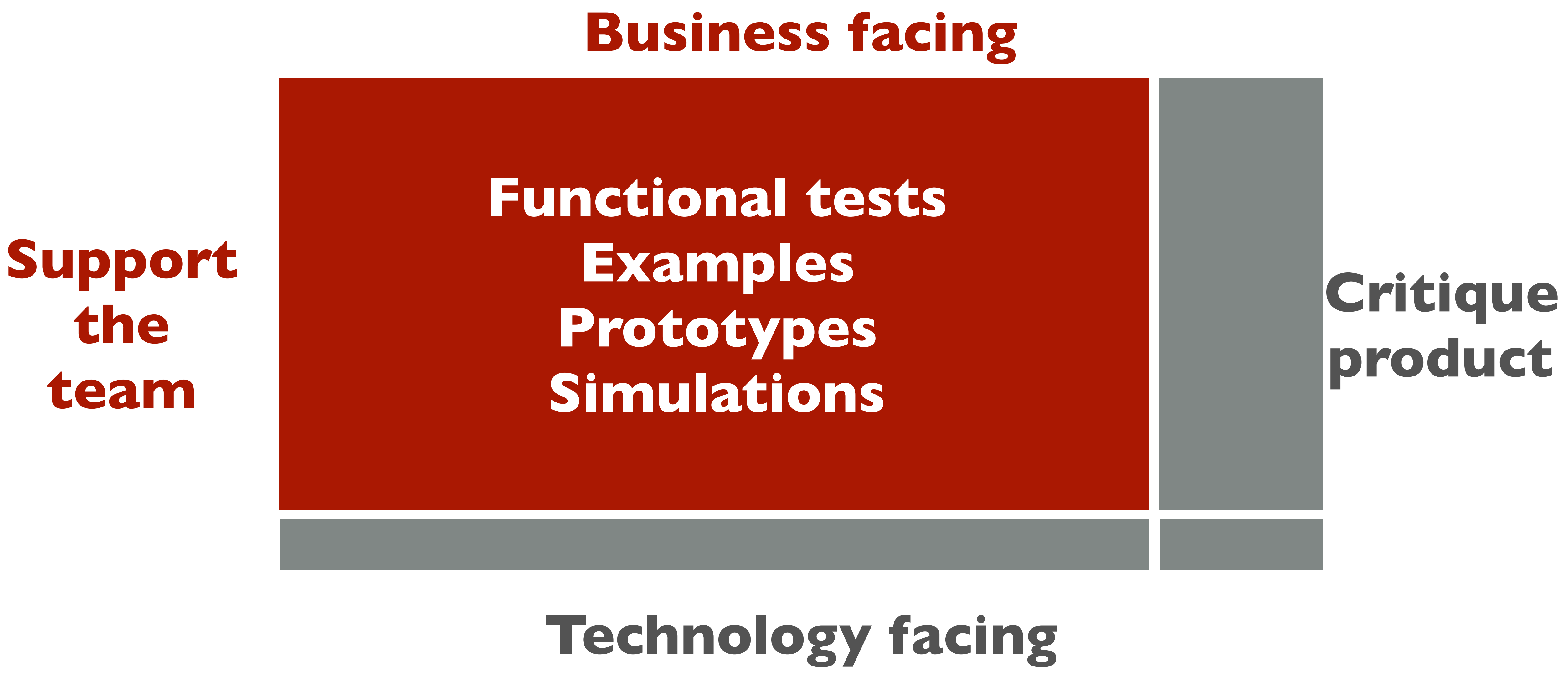
AGILE TESTING QUADRANTS: Q4



AGILE TESTING QUADRANTS



Agile testing quadrants: Q2



Functional tests

- With **functional tests**, we want to validate that the system does what it it supposed to do **from the users point of view**.
- Very often, this means **defining usage scenarios** (**test cases**). We describe the steps to be followed by users and the expected results.
- When we evaluate a software release, we can **check** whether the defined test cases can be executed with success.

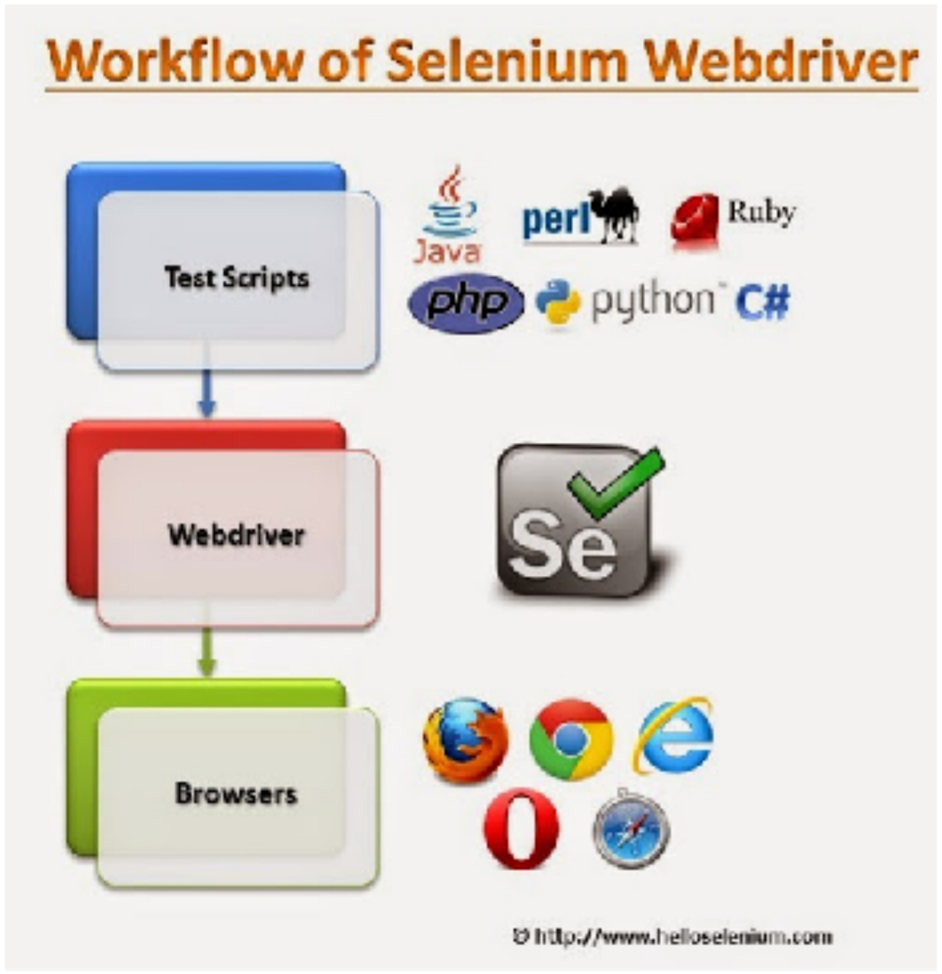
Manual functional tests

- In many organizations, test cases are documented in **test management software**. They are executed by **human operators**.
- This is a **repetitive process** with little added value.
- This is a **slow process**.
- It creates **overhead** and often gives a **false sense of confidence**.
- If you release every 3 months, it “might” be possible to do manual test campaigns. If you release on a weekly basis, it is just not possible.

Automated functional tests

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- There are now **tools** that can be used to **simulate human users**.
- With these tools, you write scripts. When the scripts are executed, they **control a web browser** and check that the content of the pages is.
- **It is not a free lunch.** Writing these scripts takes time. Maintaining these scripts (when the UI changes) takes a lot of time.
- Integration tests are slower than unit tests. Automated functional tests are **a lot slower** than integration tests.
- For this reason, they are not executed as often (at a later stage in the continuous delivery pipeline).


Workflow of Selenium Webdriver
Test Scripts
Java
perl
Ruby
php
python
C#
Webdriver
Se
Browsers
© http://www.helloselenium.com

Behaviour Driven Development (BDD)

- With Unit Tests, developers have a way to **specify and check** the behaviour of a tiny piece of code.
- The same principle can be applied with higher-level, **business oriented tests**. This is the idea of "behaviour driven development" or BDD.
- BDD is a method that **facilitates the collaboration** between business analysis, developers and testers. It gives them a **common vocabulary**.

BDD: Naming & Vocabulary Matters

- "Test method names should be sentences".
- Compare the two representations of the same "specification". It suggests that tools can support communication by emphasizing a common language for the domain.

see http://agiledox.sourceforge.net/

```
public class FooTest extends TestCase {
    public void testIsASingleton() {}
    public void testAReallyLongNameIsAGoodThing() {}
}
```

```
Foo
- is a singleton
- a really long name is a good thing
```

BDD: "Ubiquitous Language"

- BDD proposes a template to describe the intended behaviour of a system. The template is used to specify the acceptance criteria for a given user story.

Given some initial context (the givens),
When an event occurs,
then ensure some outcomes.

USER STORY
As a customer,
I want to withdraw cash from an ATM,
so that I don't have to wait in line at the bank.

ACCEPTANCE CRITERIA
Given the account is in credit
And the card is valid
And the dispenser contains cash
When the customer requests cash
Then ensure the account is debited
And ensure cash is dispensed
And ensure the card is returned

A
G
A
A
W
T
A
And ensure the card is returned

BDD: Executable Specifications

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **"Acceptance criteria should be executable"**
- We need tools that allow:
  - **analysts** to write the acceptance criteria in plain english, following the previous template;
  - **developers** to write test fixtures that act as intermediary between the specification and the system to test;
  - the **continuous delivery pipeline** to execute the specifications automatically, to integrate the test results in the "live" specification, to notify the team about the results.

Process : When will be done?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Executable Specifications

Acceptance criteria for stories are defined as scenarios.

Linking the specs with the system

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

Executable Specifications

Test Fixtures

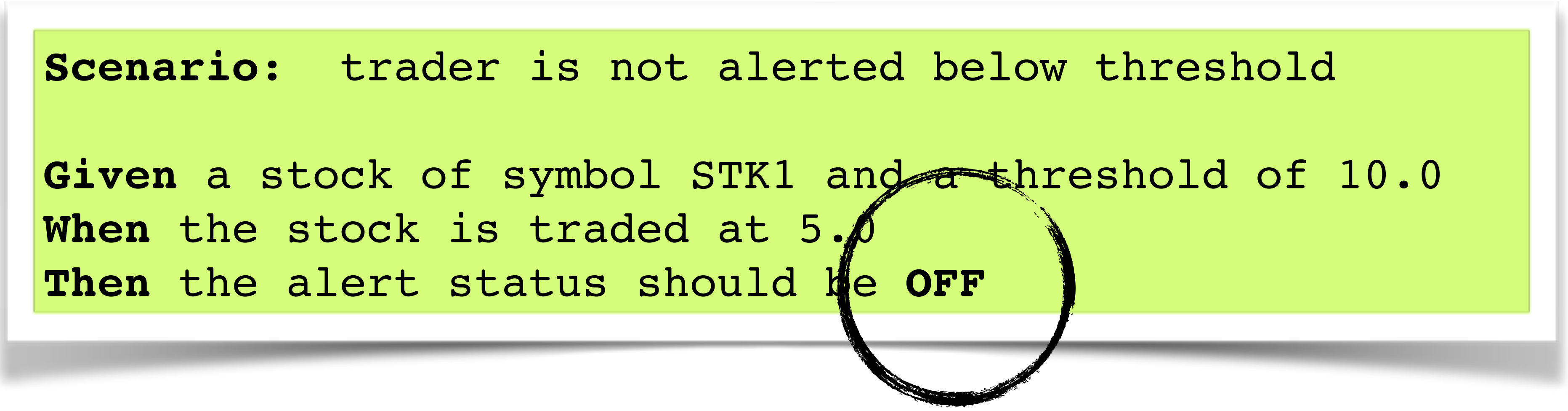
System Under Test (SUT)

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

```
public class TraderSteps { // look, Ma, I'm a POJO!!

    private Stock stock;

    @Given("a stock of symbol $symbol and a threshold
of $threshold")
    public void aStock(String symbol, double threshold)
{
        stock = new Stock(symbol, threshold);
    }

    @When("the stock is traded at $price")
    public void theStockIsTradedAt(double price) {
        stock.tradeAt(price);
    }

    @Then("the alert status should be $status")
    public void theAlertStatusShouldBe(String status) {
        ensureThat(stock.getStatus().name(),
equalTo(status));
    }

}
```

Process : let's see if we are done...

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch



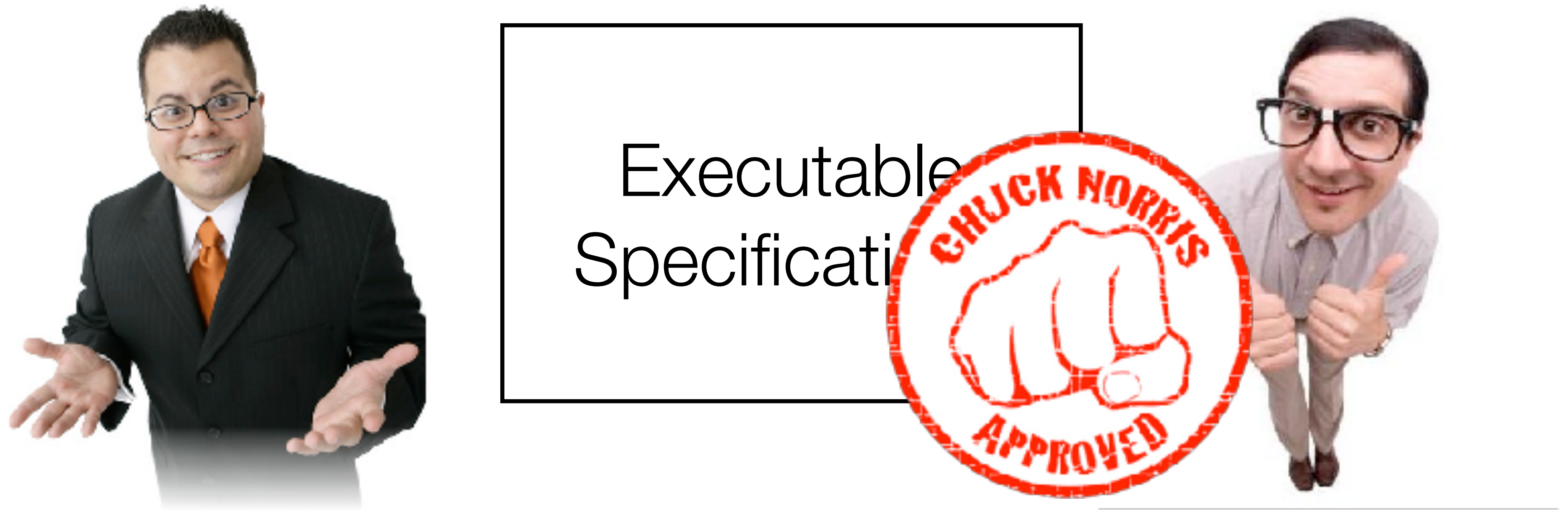
```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Executable Specifications

The test results are displayed directly in the "living" specs (other reports and notifications are also useful!)

Process : yeah!!!!!!

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```



The test results are displayed directly in the “living” specs (other reports and notifications are also useful!)

Process : noooooooooo....

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
Scenario:  trader is not alerted below threshold

Given a stock of symbol STK1 and a threshold of 10.0
When the stock is traded at 5.0
Then the alert status should be OFF
```

Executable
Specifications

The test results are displayed directly in the "living" specs (other reports and notifications are also useful!)

I can't wait to get started... what should I do?

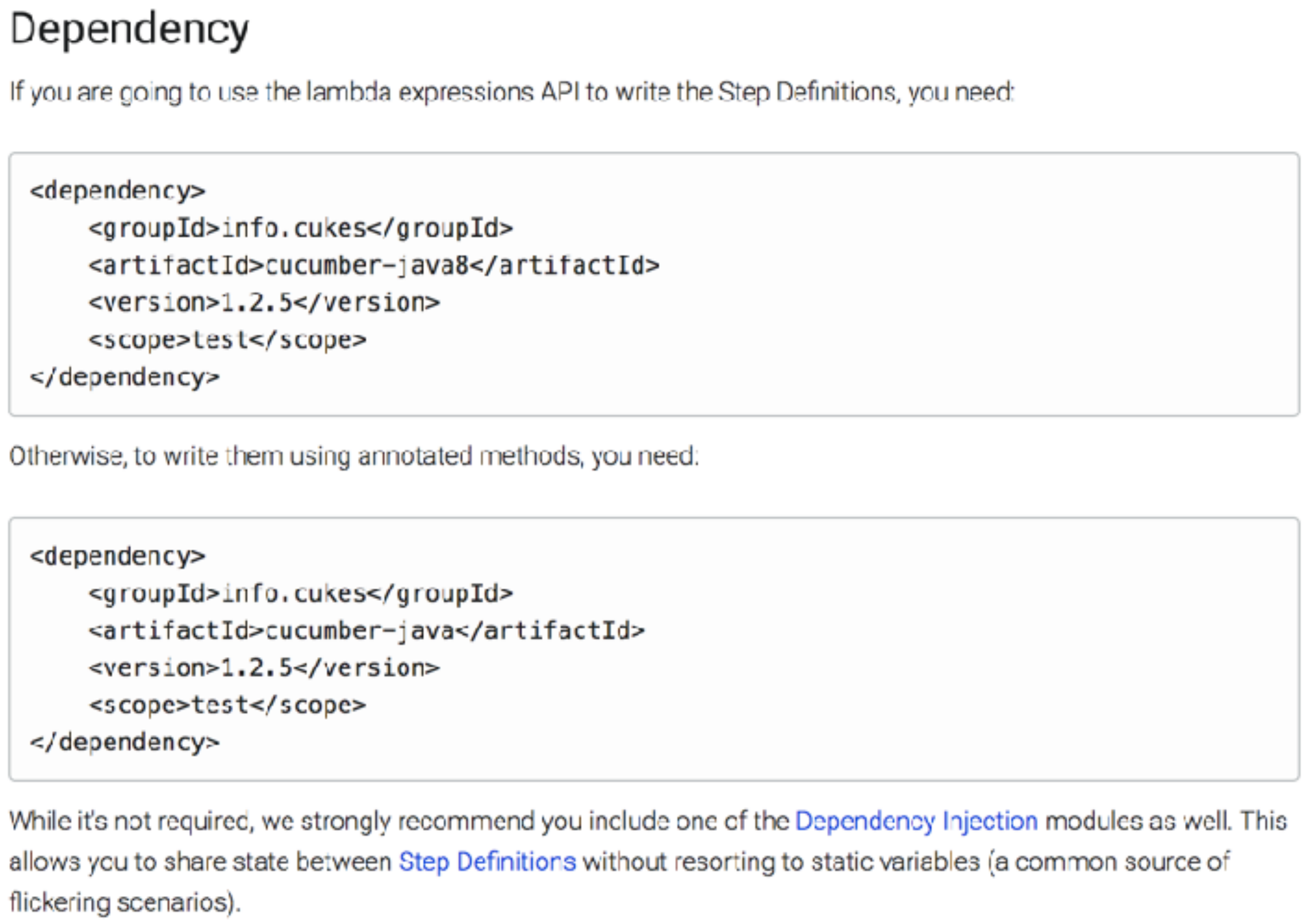
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

BDD Kickstart - Boston, US - August 2017

An open-source tool for executable specifications

A vibrant community

An ingenious company



Dependency

If you are going to use the lambda expressions API to write the Step Definitions, you need:

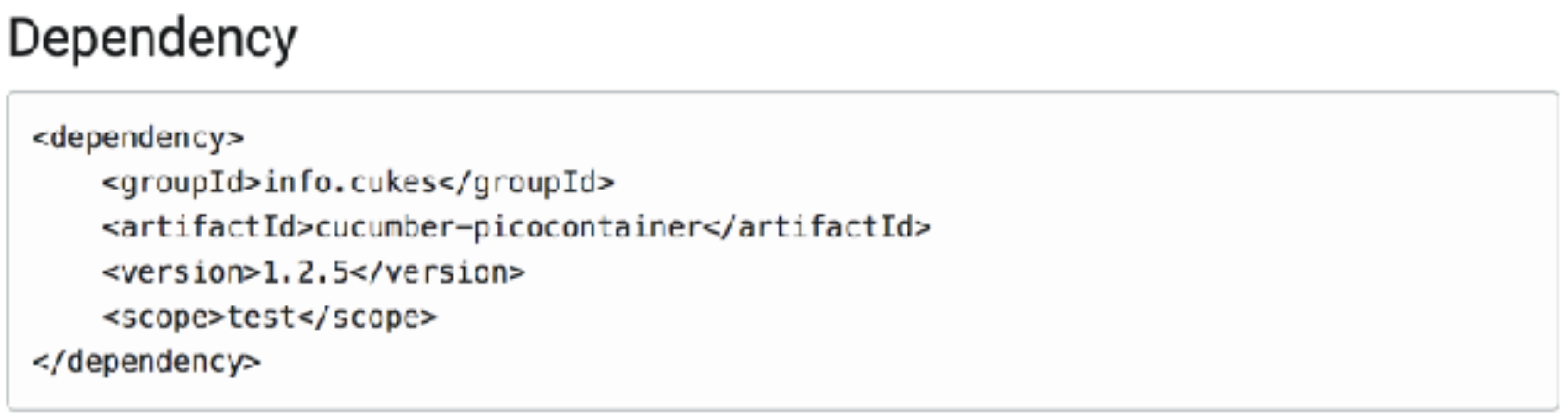
```
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-java8</artifactId>
    <version>1.2.5</version>
    <scope>test</scope>
</dependency>
```

Otherwise, to write them using annotated methods, you need:

```
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>1.2.5</version>
    <scope>test</scope>
</dependency>
```

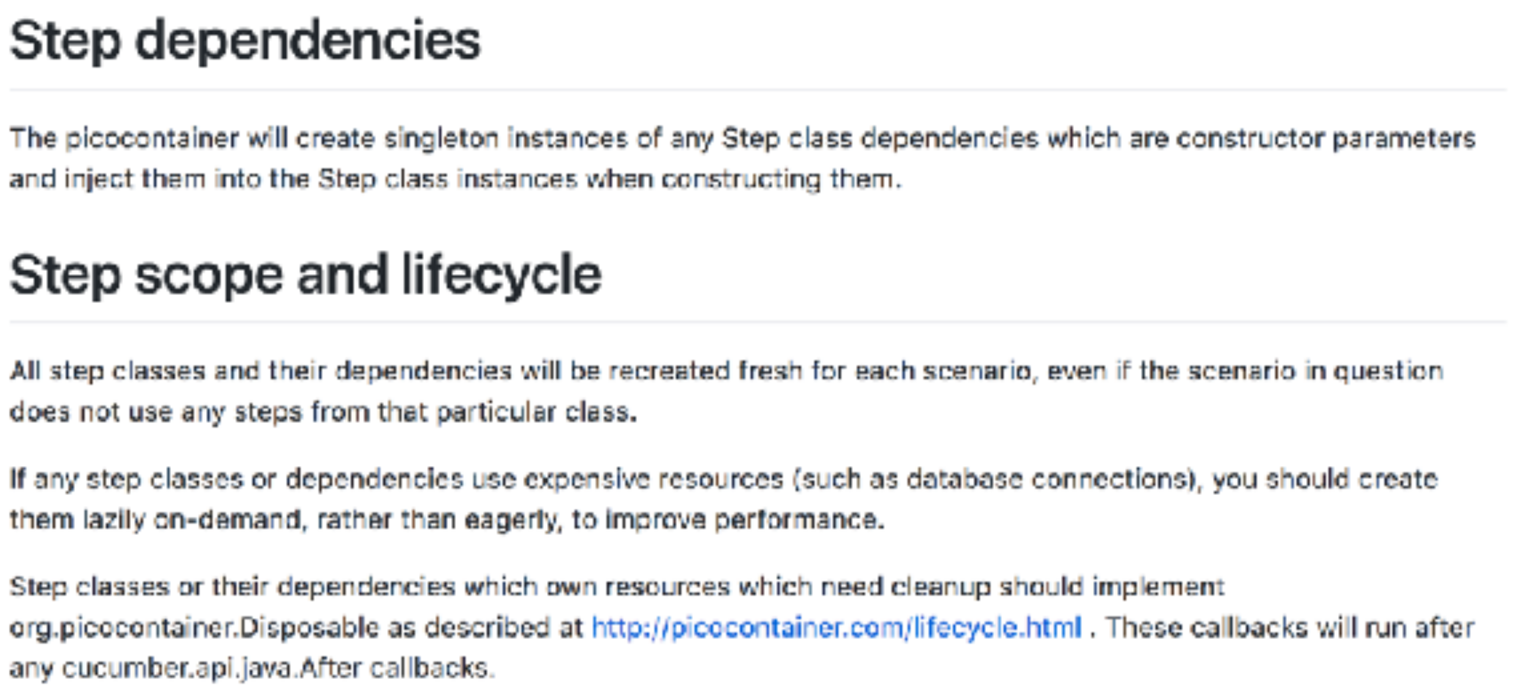
While it's not required, we strongly recommend you include one of the Dependency Injection modules as well. This allows you to share state between Step Definitions without resorting to static variables (a common source of flickering scenarios).

PicoContainer



Dependency

```
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-picocontainer</artifactId>
    <version>1.2.5</version>
    <scope>test</scope>
</dependency>
```



Step dependencies

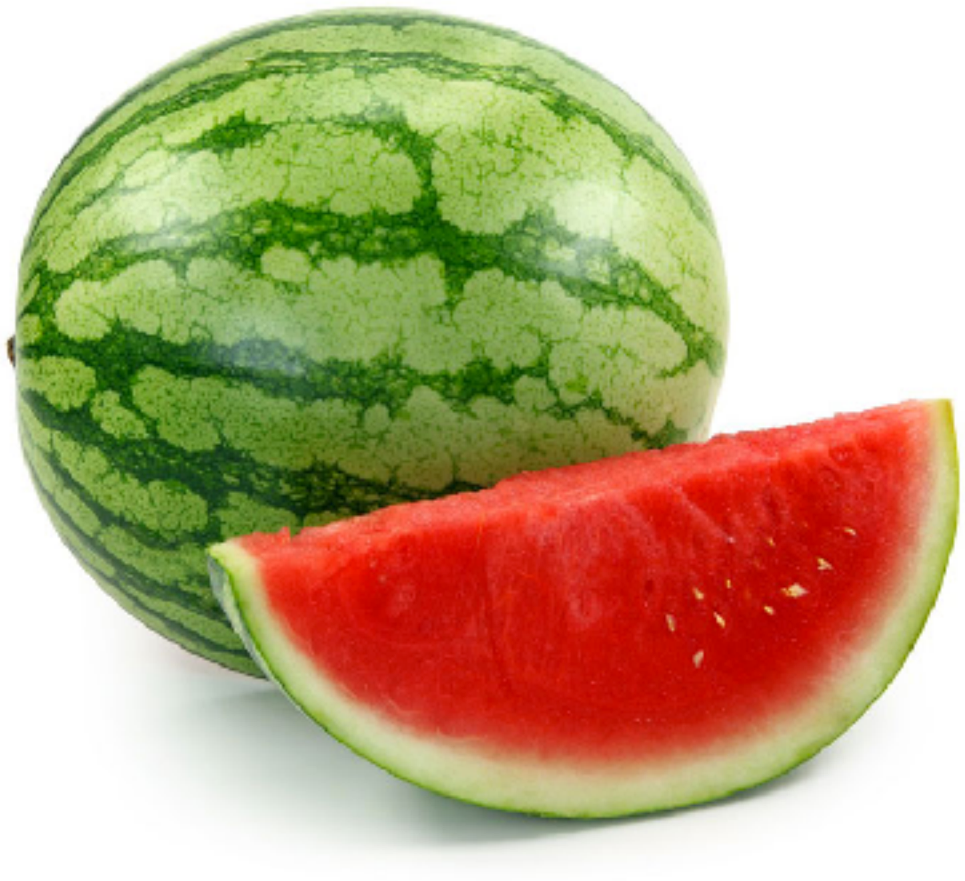
The picocontainer will create singleton instances of any Step class dependencies which are constructor parameters and inject them into the Step class instances when constructing them.

Step scope and lifecycle

All step classes and their dependencies will be recreated fresh for each scenario, even if the scenario in question does not use any steps from that particular class.

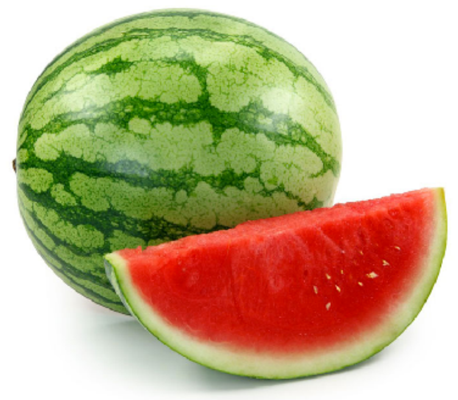
If any step classes or dependencies use expensive resources (such as database connections), you should create them lazily on-demand, rather than eagerly, to improve performance.

Step classes or their dependencies which own resources which need cleanup should implement org.picocontainer.Disposable as described at http://picocontainer.com/lifecycle.html . These callbacks will run after any cucumber.api.java.After callbacks.

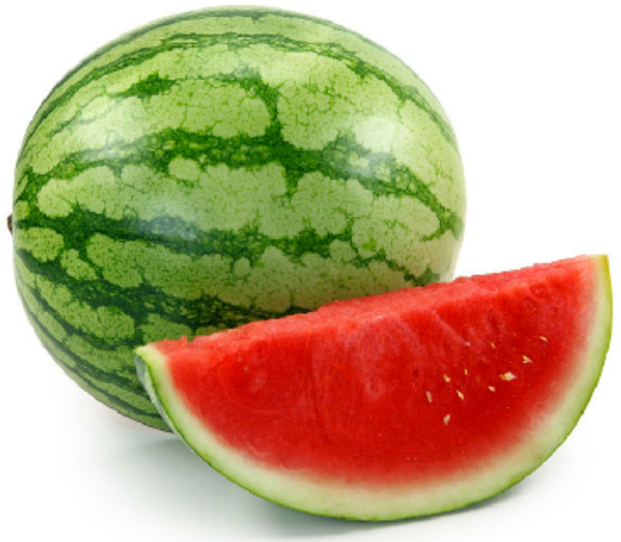


```
Feature: Creation of fruits

  Background:
    Given there is a Fruits server

  Scenario: create a fruit
    Given I have a fruit payload
    When I POST it to the /fruits endpoint
    Then I receive a 201 status code
```

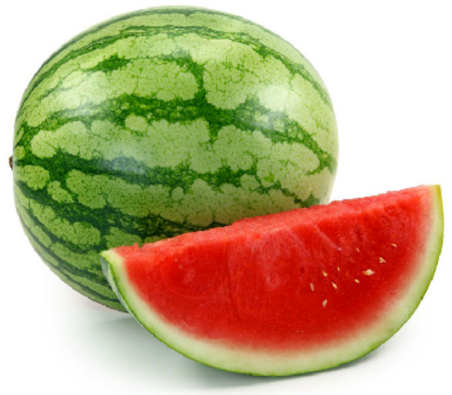
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running io.avalia.fruits.api.spec.SpecificationTest
Feature: Creation of fruits

  Background:                     # creation.feature:3
    Given there is a Fruits server

  Scenario: create a fruit                # creation.feature:6
    Given I have a fruit payload
    When I POST it to the /fruits endpoint
    Then I receive a 201 status code

1 Scenarios (1 undefined)
4 Steps (4 undefined)
0m0.000s
```



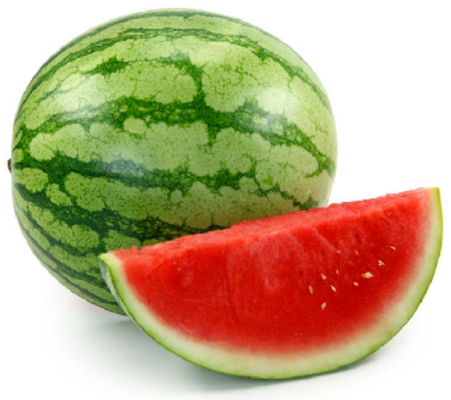
```
You can implement missing steps with the snippets below:

@Given("^there is a Fruits server$")
public void there_is_a_Fruits_server() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

...
```



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running io.avalia.fruits.api.spec.SpecificationTest
Feature: Creation of fruits

  Background:                      # creation.feature:3
    Given there is a Fruits server # CreationSteps.there_is_a_Fruits_server()
      cucumber.api.PendingException: TODO: implement me
          at
io.avalia.fruits.api.spec.steps.CreationSteps.there_is_a_Fruits_server(CreationSteps.java:16)
          at ✽.Given there is a Fruits server(creation.feature:4)


  Scenario: create a fruit                 # creation.feature:6
    Given I have a fruit payload           # CreationSteps.i_have_a_fruit_payload()
    When I POST it to the /fruits endpoint # CreationSteps.i_POST_it_to_the_fruits_endpoint()
    Then I receive a 201 status code       # CreationSteps.i_receive_a_status_code(int)

1 Scenarios (1 pending)
4 Steps (3 skipped, 1 pending)
0m0.101s
```

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```java
public class CreationSteps {

    private Environment environment;
    private DefaultApi api;
    private ApiResponse lastApiResponse;
    private ApiException lastApiException;
    private boolean lastApiCallThrewException;
    private int lastStatusCode;
    Fruit fruit;

    public CreationSteps(Environment environment) {
        this.environment = environment;
        this.api = environment.getApi();
    }

    @Given("^there is a Fruits server$")
    public void there_is_a_Fruits_server() throws Throwable
{
        assertNotNull(api);
    }

    @Given("^I have a fruit payload$")
    public void i_have_a_fruit_payload() throws Throwable {
        fruit = new io.avalia.fruits.api.dto.Fruit();
    }

    @When("^I POST it to the /fruits endpoint$")
    public void i_POST_it_to_the_fruits_endpoint() throws Throwable {
        try {
            lastApiResponse = api.createFruitWithHttpInfo(fruit);
            lastApiCallThrewException = false;
            lastApiException = null;
            lastStatusCode = lastApiResponse.getStatusCode();
        } catch (ApiException e) {
            lastApiCallThrewException = true;
            lastApiResponse = null;
            lastApiException = e;
            lastStatusCode = lastApiException.getCode();
        }

    }

    @Then("^I receive a (\\d+) status code$")
    public void i_receive_a_status_code(int arg1) throws Throwable {
        assertEquals(201, lastStatusCode);
    }

}
```



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running io.avalia.fruits.api.spec.SpecificationTest
Feature: Creation of fruits

  Background:                         # creation.feature:3
    Given there is a Fruits server # CreationSteps.there_is_a_Fruits_server()

  Scenario: create a fruit                    # creation.feature:6
    Given I have a fruit payload              # CreationSteps.i_have_a_fruit_payload()
    When I POST it to the /fruits endpoint #
CreationSteps.i_POST_it_to_the_fruits_endpoint()
    Then I receive a 201 status code          # CreationSteps.i_receive_a_status_code(int)

1 Scenarios (1 passed)
4 Steps (4 passed)
0m0.496s

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.824 sec
```

Resources

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

For **AMT 2016**, we prepared tutorials on this topic. We already used Swagger and Spring Boot. You will find 2 series of webcasts that present the setup from that year:

https://www.youtube.com/playlist?list=PLfKkysTy70Qa7tSlkbsvOrRc6Ug_c0nZz

"Swagger avec Spring Boot": **7 videos**
"Swagger et Cucumber pour des spécs exécutatables": **3 videos**

Be aware that we were still using Netbeans (which caused issues) and that since then, we have improved our setup. We will therefore do things a bit differently.

Resources (2)

In **Summer 2017**, we prepared a sample project in a GitHub repo. We will use this setup today (in these slides).

https://github.com/AvaliaSystems/TrainingREST

There are two webcasts for this project:
webcast 1
webcast 2

There are 3 feature branches in the repo, one for every phase of the tutorial.